Aggregating Falcon Signatures with LaBRADOR

Marius A. Aardal[1], Diego F. Aranha[1], Katharina Boudgoust[1,2], Sebastian Kolby[1], and Akira Takahashi[3]⋆

[1] Aarhus University, Denmark
[2] CNRS, Univ Montpellier, LIRMM
[3] J.P.Morgan AI Research & AlgoCRYPT CoE

February 23, 2024

Abstract. Several prior works have suggested to use non-interactive arguments of knowledge with short proofs to aggregate signatures of Falcon, which is part of the first post-quantum signatures selected for standardization by NIST. Especially LaBRADOR, based on standard structured lattice assumptions and published at CRYPTO'23, seems promising to realize this task. However, no prior work has tackled this idea in a rigorous way. In this paper, we thoroughly prove how to aggregate Falcon signatures using LaBRADOR. First, we improve LaBRADOR by moving from a low-splitting to a high-splitting ring, allowing for faster computations. This modification leads to some additional technical challenges for proving the knowledge soundness of LaBRADOR. Moreover, we provide the first complete knowledge soundness analysis for the *non-interactive* version of LaBRADOR. Here, the multi-round and recursive nature of LaBRADOR requires a complex and thorough analysis. For this purpose, we introduce the notion of *predicate special soundness (PSS)*. This is a general framework for evaluating the knowledge error of complex Fiat-Shamir arguments of knowledge protocols in a modular fashion, which we believe to be of independent interest. Lastly, we explain the exact steps to take in order to adapt the LaBRADOR proof system for aggregating Falcon signatures and provide concrete estimates for proof sizes. Additionally, we formalize the folklore approach of obtaining aggregate signatures from the class of hash-then-sign signatures through arguments of knowledge.

⋆ Work partially done while affiliated with the University of Edinburgh.

Table of Contents

1 Introduction

In 2022, the US National Institute of Standards and Technology (NIST) announced the first protocols, deemed secure even in the presence of quantum computers, for standardization.[4] Falcon [PFH+22], whose security relies on structured lattice problems, is one of the three signature protocols selected by NIST. A natural question now is whether Falcon can be used in more advanced cryptographic settings. In this work, we study the question of *aggregating* many Falcon signatures into a single one.

Aggregate signatures (AS), introduced by [BGLS03], allow to combine N individual signatures, on possibly distinct messages and public keys, into one aggregated signature σ_{agg}. This feature is beneficial whenever a large amount of signatures have to be sent and bandwidth is a bottleneck. It gained a lot of attention in the past few years as aggregate signatures are used on a large scale in blockchains. As an example, Ethereum 2 is currently using aggregate signatures based on pairings [BDN18], as detailed in the annotated specifications.[5] Preferably, the aggregation does not require the interaction of the signing parties. This is desirable especially if many parties are involved. As many of the currently deployed cryptographic protocols, and in particular pairing-based protocols, get insecure in the presence of large scale quantum computers, it is an important research question to search for presumably quantum-resistant solutions. Lattice-based cryptography has been shown to be one of the most promising directions.

Recently, there have been multiple proposals of aggregating lattice-based signatures, cf. Section A for a detailed related work discussion. However, the only known AS tailored to Falcon are *sequential* and thus require some form of interaction between signers [EB14, WW19]. Although [WW19] explicitly instantiates their scheme with Falcon, it turned out to be susceptible to a simple forgery attack [BT23]. Moreover, the size of an aggregate signature is still linear in the number N of signatures involved, and [BT23] reports the compression rate (i.e. the size of AS divided by the size of N signatures) is only about 60% even if the GPV-based scheme of [EB14] is adapted to Falcon.

When it is challenging to design an AS tailored to a specific signature scheme, it is a natural question to investigate whether *generic* solutions exist. One generic approach for aggregation proposed in the literature is to use non-interactive arguments of knowledge (AoK) [ACL+22, DGKV22, WW22].[6] Given N signatures issued for possibly distinct public keys and messages, we set as witness w the signatures and as statement x the corresponding public keys and messages, i.e., i.e., $\mathsf{w} := \{\sigma_i\}_{i\in[N]}$ and $\mathsf{x} := \{\mathsf{pk}_i, m_i\}_{i\in[N]}$. The signature scheme defines the binary relation $R = \left\{(\mathsf{x}, \mathsf{w}) \,:\, \mathsf{Ver}(\mathsf{pk}_i, m_i, \sigma_i) = 1 \ \forall i \in [N]\right\}$, where $\mathsf{Ver}(\cdot,\cdot,\cdot)$ is the verification algorithm of the signature. Any non-interactive argument of knowledge Π for an NP language can be used to produce for a statement x a proof π of a corresponding witness w fulfilling the binary relation R. In particular, if Π is *succinct* (i.e. its proof size is at most polylogarithmic in the size of the statement and witness, henceforth SNARK) [ACL+22] or a rate-1 *batch* argument (i.e. its proof size is independent of N the number of NP statements, henceforth BARG) [DGKV22, WW22], one can construct a compact AS scheme by setting the proof as an aggregate signature.

Concretely, the idea of aggregating Falcon using SNARKs was sketched in a recent lattice-based SNARK [ACL+22, Sec. 7.2]. In particular, they observed that the verification equation of Falcon can be expressed in the native language of their SNARK (i.e. without converting the equation into general circuit constraints). However, they left it open to rigorously realize the idea. Moreover, their proof system relies on a non-standard lattice-based knowledge assumption, which was recently broken by [WW23].

With the recent introduction of the lattice-based SNARK LaBRADOR [BS23], a significantly more efficient proof system whose security relies on standard structured lattice problems was proposed. Further, the native language of LaBRADOR seems even more suitable for aggregating Falcon signatures. Again, the idea of using LaBRADOR to aggregate signatures was sketched in [TS23] without providing security proofs and concrete estimates for the computation times and proof sizes for any particular signature scheme. Moreover, their approach requires translating the verification conditions of the signature scheme into R1CS, which does not natively support the relation for batch-proving Falcon signatures. Given all this, we are motivated to ask the following question:

Can LaBRADOR be used to aggregate Falcon signatures while providing (1) a rigorous security proof, and (2) concrete estimates?

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

[5] https://github.com/ethereum/annotated-spec/blob/master/phase0/beacon-chain.md

[6] To be more precise, only a relaxed version of knowledge soundness called *somewhere extractability* is sufficient for constructing AS.

1.1 Our Contributions

In this work, we give a positive answer to this question. Along the way, we develop a number of technical tools that we believe to be of independent interest. First, we improve LaBRADOR by moving from a low-splitting to a high-splitting ring, allowing for faster computations. This modification leads to some additional technical obstacles for proving the knowledge soundness of LaBRADOR. In particular, moving to a high-splitting ring affects the so-called challenge space. In Section 3 and D, we introduce a new notion of *well-spread* challenge spaces and a variant of the Schwartz-Zippel lemma. We then observe that soundness analysis of a non-interactive version of LaBRADOR is not covered by existing frameworks, such as [AFK22], due to its extensive use of probabilistic tests that cannot be captured by a simple rewinding process. In Section 4, G and H, we propose a new soundness notion for multi-round public-coin protocols, called *predicate special soundness (PSS)*. Our PSS notion leads to a more general framework for analyzing knowledge soundness of the Fiat-Shamir transform [FS87], implying the first complete knowledge soundness analysis for the *non-interactive* version of LaBRADOR in the random oracle model (Section 4 and I). We highlight Section 4 as a versatile toolkit allowing future protocols designers to derive concrete knowledge error of (possibly more complex) Fiat-Shamir AoK protocols in a modular fashion. To complete the construction of aggregate signatures, we explain in Section 5 and F the concrete steps required in order to adapt the LaBRADOR proof system for aggregating Falcon signatures. As the modulus q used in the Falcon signature scheme is optimized to be as small as possible, it does not give enough room to use LaBRADOR. Thus, we have to use another larger modulus q' in the proof system, requiring additional checks to guarantee that no wrap-around was caused and that the norms are in the right bounds. Moreover, Falcon operates on rings with large degrees. By moving to subrings of smaller degrees when instantiating LaBRADOR, we significantly reduce the proof sizes, and hence the AS sizes. Lastly, in Section 6 and E we provide concrete estimates for proof sizes and detailed comparison with other lattice-based AS. As a side contribution, we formalize the folklore approach of obtaining aggregate signatures from the class of hash-then-sign signatures through SNARKs (Appendix C).

1.2 Technical Overview

Falcon is an instantiation of the GPV framework [GPV08] for lattice-based hash-then-sign signatures over the NTRU [HPS98] class of structured lattices. It works over a power-of-two cyclotomic ring $\mathcal{R}$ modulo q, denoted by $\mathcal{R}_q$. Let us quickly recap the hash-then-sign paradigm. A key pair consists of pk defining a *preimage sampleable function (PSF)* [GPV08] $F_{\mathsf{pk}} : \mathsf{Do} \to \mathsf{Ra}$, and sk that allows one to invert F_{pk}. Upon receiving a message m to be signed, the signer first generates its hash $y = \mathsf{H}(r, m) \in \mathsf{Ra}$, where $r \in \{0,1\}^k$ is a freshly sampled random salt, and then use sk to sample a signature $\sigma = x \in \mathsf{Do}$ following some distribution $\mathscr{D}(F_{\mathsf{pk}}^{-1}(y))$. If instantiated with Falcon, $\mathsf{pk} = \boldsymbol{h} \in \mathcal{R}_q$ defines an NTRU-module $\varLambda = \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{R}^2 : \boldsymbol{u} + \boldsymbol{h}\boldsymbol{v} = \mathbf{0} \bmod q\}$ and sk contains a secret (short) basis of $\varLambda$ that allows sampling module elements following a discrete Gaussian distribution defined over an arbitrary coset $\varLambda_{\boldsymbol{t}} = \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{R}^2 : \boldsymbol{u} + \boldsymbol{h}\boldsymbol{v} = \boldsymbol{t} \bmod q\}$. Thus, the signing algorithm of Falcon first hashes m to $y = \boldsymbol{t} \in \mathcal{R}_q$, uses the secret basis to obtain a preimage $x = (\boldsymbol{s}_1, \boldsymbol{s}_2) \in \varLambda_{\boldsymbol{t}}$, and outputs $\sigma = (\boldsymbol{s}_1, \boldsymbol{s}_2, r)$ as a signature. The verification conditions are simply (1) $\boldsymbol{s}_1 + \boldsymbol{h}\boldsymbol{s}_2 = \mathsf{H}(r, m) \bmod q$, and (2) $\|(\boldsymbol{s}_1, \boldsymbol{s}_2)\|_2 \le \beta \ll q$, where β is determined by a Gaussian parameter.

Aggregation of Falcon signatures amounts to batch-proving knowledge of N Gaussian samples $(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2})_{i\in[N]}$ and salts $(r_i)_{i\in[N]}$ meeting the above verification conditions w.r.t. a list $(m_i, \boldsymbol{h}_i)_{i\in[N]}$ of (potentially distinct) messages and public keys, respectively. However, generating proof of correct hash computation is not only costly, but also leads to heuristic security guarantees: an aggregator may need a concrete description of H as a hash function, while Falcon has only been proven secure if H is modeled as a random oracle.[7] We therefore opt to let the aggregator include salts r_i in the aggregated signature and generate a proof for $\boldsymbol{t}_i = \boldsymbol{s}_{i,1} + \boldsymbol{h}_i\boldsymbol{s}_{i,2} \bmod q$ for a public statement $\boldsymbol{t}_i$, and let the verifier compute $\boldsymbol{t}_i = \mathsf{H}(r_i, m_i)$ locally. Although this approach sacrifices the asymptotic compactness of the resulting aggregate signature, the size of salt r_i is much smaller than $(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2})$ in practical parameter regimes. We empirically show that aggregating $(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2})_{i\in[N]}$ already reduces the size of signature significantly compared to the naive concatenation of N Falcon signatures. To realize an asymptotically compact scheme, our approach can easily be adapted to a deterministic variant of Falcon where there is no salt.

Adapting LaBRADOR for Aggregating Falcon Signatures. We instantiate the SNARK with LaBRADOR, a highly efficient sublinear argument based on Module-SIS. We first recall the *principal*

[7] Analogously, the security proofs for IVC [Val08] and PCD [BCMS20] often face the same problem.

relation for which LaBRADOR has been designed. The relation R consists of witness vectors $\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r \in \mathcal{R}_q^n$ and a statement $\mathsf{x} = (\mathcal{F}, \beta)$, where $\mathcal{F}$ is a collection of dot product constraints and β is a norm bound check, over $\mathcal{R}_q$ and $\mathcal{R}$ respectively. We call n the *rank* and r the *multiplicity* of the witnesses. Each dot product constraint $f \in \mathcal{F}$ is defined by a function of the form

$$f(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) := \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_i \rangle - \boldsymbol{b},$$

where $\boldsymbol{a}_{i,j}, \boldsymbol{b} \in \mathcal{R}_q$ and $\vec{\boldsymbol{\varphi}}_i \in \mathcal{R}_q^n$, such that $\boldsymbol{a}_{i,j} = \boldsymbol{a}_{j,i}$ for all i, j. We say that the principal relation is satisfied if $\forall f \in \mathcal{F}, f(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) = \boldsymbol{0}$ and $\sum_{i=1}^{r} \|\vec{\boldsymbol{w}}_i\|_2^2 \leq \beta^2$.

At first glance, the verification equation and norm bound of a single Falcon signature seem quite compatible with the principal relation of LaBRADOR. To aggregate N signatures, one might then try to extend the statement to contain a verification equation for each signature and a combined norm bound: $\forall i = 1, \ldots, N, \boldsymbol{s}_{i,1} + \boldsymbol{h}_i \boldsymbol{s}_{i,2} - \boldsymbol{t}_i = \boldsymbol{0} \bmod q$ and $\sum_{i=1}^{N} \sum_{j=1}^{2} \|\boldsymbol{s}_{i,j}\|_2^2 \leq N\beta^2$. If LaBRADOR was instantiated over the ring used by Falcon then $(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2})_{i \in [N]}$ could be used directly as witness vector of multiplicity $r = 2N$ of rank $n = 1$ in $\mathcal{R}_q^n$. However, there are several problems with this approach.

The first problem is that the norm check in LaBRADOR is both approximate and with respect to the entire witness. This introduces a degree of slack that grows with the number of signatures N. When reducing the unforgeability of our aggregate signature scheme to the unforgeability of Falcon, we need that the knowledge extractor outputs a witness consisting of valid Falcon signatures (see details of generic AS construction from SNARK in Appendix C). In particular, we need to be able to guarantee that they have ℓ_2-norm at most β. In Subsection 5.2, we therefore modify the first iteration of the LaBRADOR protocol to use the approach of [GHL22] for an exact proof of smallness. The norm checks in the subsequent iterations are only for the binding of the commitments, so no modifications have to be made there.

The second problem we encounter appears when we look closer at how exactly LaBRADOR performs its consolidated norm check. Recall, to verify the norm of the witness LaBRADOR uses an (approximately) distance preserving projection to compute a much smaller vector, to send to the prover. Specifically, the modular Johnson-Lindenstrauss projections (see Lemma 2.2) are used, which require that the norm bound b of the statement satisfies the inequality $\sqrt{\lambda} b \leq q/C_1$ for security level λ and some corresponding constant C_1. For both Falcon parameter sets this is not satisfied, even when restricting b to the norm of just a single signature. Therefore, if we wish to use the Johnson-Lindenstrauss projection, we need a larger modulus. In Section 5.1, we reformulate the statement and witness so that the LaBRADOR protocol uses a separate modulus q', different from q.

The last problem with this formulation is that the number of initial witness elements $r = 2N$ (actually, r is even bigger due to the aforementioned modifications) and their rank $n = 1$ is quite unbalanced. For the performance of LaBRADOR, the relation between the multiplicity r and the rank n is important (cf. Section 2.4). In Section 5.3, we present an alternative formulation of the constraints that achieves a better balance between these parameters. This new formulation gives us better runtimes for the prover and verifier and slightly shorter aggregation proofs. For a full list of the final set of constraints, see Appendix F.2.

In a first instantiation of LaBRADOR with the constraint system described above, we maintained the same ring $\mathcal{R}$ as for Falcon. However, we obtained surprisingly large proof sizes and found out that the large ring degrees of Falcon $d \in \{512, 1024\}$ were the main reason for this. As detailed out in Section 5.4, we were able to significantly compress the proof sizes by moving to subrings $\mathcal{S}$ of smaller degrees d'.

Choice of Ring and Challenge Space. As explained before, Falcon operates over a power-of-two cyclotomic ring $\mathcal{R}$ of degree d modulo q, while our modified LaBRADOR operates over $\mathcal{S}$ of degree d' modulo q'. For a given degree d', we can vary the modulus q' in order to obtain different mathematical properties of $\mathcal{S}_{q'}$. In particular, the relation between d' and q' defines how well $\mathcal{S}$ is *splitting* into *CRT-slots*. Without going into the mathematical details here, we mainly distinguish two settings: low-splitting (with few CRT-slots) and high-splitting (with many CRT-slots) regimes. When invoking lattice-based SNARKs, one makes use of a subset $\mathcal{C} \subset \mathcal{S}_{q'}$, which is called the challenge space.

LaBRADOR [BS23] opted for a low-splitting regime which allows to design a simple yet useful challenge space $\mathcal{C}$ for their protocol. In particular, they exploit the facts that in the low-splitting regime 1) every non-zero element of small enough norm is invertible [LS18] and 2) the size of each of the few CRT-slots is exponentially big in the ring degree d'. However, low-splitting rings do not allow for fast computations as they do not benefit from the nice properties of the number theoretic transform.

We propose an improvement of LaBRADOR allowing for much faster computations by moving from low-splitting to high-splitting rings with only small losses in proof sizes. This comes with some additional technical challenges as we detail out in the following. First and foremost, not every small enough ring element is still invertible. By designing suitable challenge spaces, however, one can show that the probability that a challenge element (and the difference of two distinct challenges) is non-invertible is as small as $2^{-\lambda}$ for a targeted security level λ [ALS20, ESZ22, ESLR23]. The previous approaches have been quite ad-hoc and we think it is of independent interest to abstract away the needed property of $\mathcal{C}$ to prove knowledge soundness of most of the lattice-based SNARKs, and in particular of LaBRADOR. To this end, we introduce the concept of *well-spread* challenge spaces in Section 3.1. At a high level, well-spreadness of $\mathcal{C}$ directly links the splitting behavior of $\mathcal{S}_{q'}$ to the probability of invertibility of elements in $\mathcal{C}$.

LaBRADOR (implicitly) also used the fact that every small enough ring element is invertible in the two-splitting regime in order to apply a variant of the Schwartz-Zippel lemma over low-splitting rings [BCPS18]. In Section 3.2, we prove the first generalization of Schwartz-Zippel to the high-splitting regime by connecting it to the previously introduced well-spreadness. This might be relevant for other future lattice-based SNARKs. Lastly, we show in Section D.4 that challenge sets previously proposed for the high-splitting setting in the literature [ESLR23] fulfill well-spreadness and use them to instantiate our challenge set for aggregating Falcon signatures through LaBRADOR.

The splitting regime has also an impact on how small the degree d' of the subring $\mathcal{S}$, to which we move, can be. As in the original LaBRADOR specification, for security level roughly $\lambda = 128$ and in the low-splitting regime, the optimal degree to choose is $d' = 64$. When moving to the high-splitting regime, for the same security level, we can only achieve degree $d' = 128$. Note that the increase of the subring degree leads to noticeably larger proof sizes, but the gain in computation efficiency when moving from low-splitting to high-splitting rings is by far more important. We thus think that overall the high-splitting regime is more interesting.

Soundness Analysis of Non-Interactive LaBRADOR. LaBRADOR is a multi-round public-coin protocol with a structure partially resembling Bulletproofs-style [BCC⁺16, BBB⁺18] recursive protocols. Attema *et al.* [AFK22][8] recently proved that $(2\mu+1)$-round interactive protocols with $\mathbf{K} = (k_1, \ldots, k_\mu)$-tree special soundness give rise to Fiat-Shamir non-interactive AoK with knowledge error $\kappa \in O(Q \cdot \kappa')$ in the random oracle model, where Q is the number of RO queries made by a cheating prover and κ' is a knowledge error of the underlying interactive protocol. Unfortunately, the result of [AFK22] doesn't immediately allow us to derive a concrete knowledge error for Fiat-Shamir LaBRADOR because (interactive) LaBRADOR doesn't satisfy the $\mathbf{K}$-tree special soundness. There are three main technical hurdles when adapting AFK: (1) If the challenge space is imperfect (i.e. not every challenge difference is invertible in $\mathcal{R}_{q'}$), a tree with edges labeled by distinct challenges does not necessarily allow for extraction. Thus, we must take into account the probability that the extractor fails by hitting a bad challenge. (2) AFK only covers the case where extraction of a valid witness (or a solution to some computational problem) is always successful once a tree of accepting transcripts is given, whereas in LaBRADOR only a candidate witness is obtained and to check its validity the extractor must additionally perform probabilistic checks w.r.t. this fixed candidate, using a freshly sampled challenge from other rounds. (3) AFK only considers a tree of transcripts where each node is labeled by the prover's message, and each edge is labeled by a single challenge value. Its requirement of successful extraction is simply that the edges linked to the same node have distinct labels, whereas to extract a (candidate) witness in LaBRADOR, one needs a tree of transcripts where each edge is labeled by a *vector* of challenges in the amortization round and those vectors have to be distinct coordinate-wise.

To resolve the issues (1) and (2) altogether, in Section 4 we extend the $\mathbf{K}$-tree special soundness notion with predicate system $\boldsymbol{\Phi}$, dubbed $(\boldsymbol{K}, \boldsymbol{\Phi})$*-predicate-special-soundness (PSS)*. On a high-level, $\boldsymbol{\Phi}$ is a collection of predicates defined for every level of a given $\mathbf{K}$-tree of transcripts, describing "well-formedness" of sub-trees bottom-up. We then consider two types of predicates: *challenge predicates* which enforce special properties on the challenges for sibling nodes, and *commitment predicates* which enforce properties on commited values for sub-trees, helping to extract a valid witness. We then define *failure density* for every predicate to bound the number of bad challenges for an arbitrary fixed context, and use failure density to derive a knowledge error of Fiat-Shamir-transformed $(\boldsymbol{K}, \boldsymbol{\Phi})$-PSS protocols (formally stated in Theorem 4.1). Like in the base result of AFK, we conclude that the concrete knowledge error is still linear in Q. To illustrate the usefulness of our PSS in an accessible manner, we provide analysis of a bare-bones version of LaBRADOR in Section 4. Finally, to address the issue (3) we further generalize the PSS notion by incorporating *coordinate-wise special soundness (CWSS)* of Fenzi and Nguyen [FMN23] in Section G.

Estimates. Putting the aforementioned techniques together, we are able to provide parameter sets for our AS instantiated with optimized LaBRADOR and Falcon-512 or Falcon-1024. In Section 6 and Appendix E,

[8] In an independent and concurrent work Wikström achieved a similar result, using alternate techniques [Wik21]. We will focus on the result of Attema *et al.* as we take departure from their approach. In Section A.2 we compare our approach to other related works on concrete analysis of Fiat-Shamir.

we compare our AS with (1) the trivial concatenation of N Falcon signatures, (2) the GPV-based AS of [JRS23], and (3) the Merkle-tree-based AS of [FSZ22, FHSZ23]. fullversion We provide our program code in the repository at `https://github.com/dfaranha/aggregate-falcon`. We observe that for both parameter regimes starting from ca. 200 signatures, our AS is shorter than the trivial solution. For example, if $N = 2000$, our AS achieves a compression rate of less than 18%. For $N = 8192$, it even goes down to less than 9%. We also show that our AS is significantly shorter than [JRS23] and [FSZ22]. Although [FHSZ23] outputs slightly smaller aggregate signatures, our approach has advantages in that it is provable secure under the standard security notion for AS (in contrast to the "synchronized" model of [FSZ22, FHSZ23]), and that it is compatible with the standardized Falcon scheme. We also provide concrete estimates of computation times for polynomial arithmetic, and confirm that our adapted LaBRADOR compatible with almost-fully-splitting rings would significantly improve the performance of the prover and verifier.

From SNARKs to Aggregate Signatures. Although it might be tempting to conclude the security of SNARK-based aggregate signatures AS assuming knowledge soundness of the argument system Π and $\mathsf{EU\text{-}CMA}$ security of the signature scheme S, there is a subtle gap that was already pointed out in a different context [FN16]. Intuitively, the reduction to $\mathsf{EU\text{-}CMA}$ of S proceeds as follows. If an AS adversary $\mathcal{A}$ against AS outputs a valid aggregate signature (i.e. proof) π, by the knowledge soundness assumption, there should exist an efficient extractor $\mathcal{E}$ that can extract from $\mathcal{A}$ a valid signature σ (i.e. witness) w.r.t. a challenge public key pk. Thus, a reduction $\mathcal{B}$ may internally invoke $\mathcal{E}$ against $\mathcal{A}$ in order to obtain a signature with which $\mathcal{B}$ breaks the $\mathsf{EU\text{-}CMA}$ game. However, notice that the usual knowledge soundness definition does not guarantee successful witness extraction from a cheating prover if given access to an additional oracle, even though an AS adversary $\mathcal{A}$ may attempt to produce a malicious proof after adaptively querying the signing oracle (see Game 2 for the formal definition of security notion). Fiore and Nitulescu [FN16] in fact showed the existence of powerful signing oracles that completely undermine knowledge soundness for *any* SNARK (in the standard model), assuming universal one-way hash functions. This means that, one cannot prove the security of generic SNARK-based AS from arbitrary Π and S; instead, one has to prove that a certain class of signing oracles do not interfere with witness extraction for Π and thus successful reduction to $\mathsf{EU\text{-}CMA}$ security of S. To this end, we extend one of the positive results of Fiore-Nitulescu to prove that the reduction $\mathcal{B}$ indeed succeeds even if $\mathcal{A}$ has access to a signing oracle for (both salted and deterministic) hash-then-sign-type schemes. In our theorem, the only additional requirement for Π is that its knowledge soundness holds against an adversary receiving auxiliary input consisting of random elements in the PSF range and their corresponding preimages, which are rather mild since they are generated independently of the CRS and/or random oracle used by an argument system Π. Our general results in Appendix C are not tailored to specific instantiations of hash-then-sign and argument systems, and thus they may be of independent interest for future designers of SNARK-based AS constructions.

2 Preliminaries

2.1 Notation

Let q be an odd prime and d a power of 2. The $2d$-th cyclotomic ring is defined as $\mathcal{R} = \mathbb{Z}[X]/\langle X^d + 1\rangle$. Throughout this paper, we work over $\mathcal{R}$ modulo q, denoted by $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$. We use bold letters for polynomials $\boldsymbol{r} \in \mathcal{R}_q$ to differentiate them from integers $r \in \mathbb{Z}_q$. This pattern extends to vectors $\vec{\boldsymbol{v}} \in \mathcal{R}_q^n$, $\vec{v} \in \mathbb{Z}_q^n$ and matrices $\boldsymbol{A} \in \mathcal{R}_q^{m\times n}$, $A \in \mathbb{Z}_q^{m\times n}$. For $\boldsymbol{r} = \sum_{i=0}^{d-1} r_i X^i \in \mathcal{R}_q$, we use $\mathsf{ct}(\boldsymbol{r})$ to denote its constant coefficient r_0. Vector concatenation is denoted $\vec{v}_1 \| \vec{v}_2$. Several norms are of interest in this paper. For $\vec{v} \in \mathbb{Z}_q^n$ and $p \in \{1, 2, \infty\}$, we define $\|\vec{v}\|_p$ as the ℓ_p-norm of its unique representative in $[\pm\frac{q-1}{2}]^n := [-\frac{q-1}{2}, \ldots, +\frac{q-1}{2}]^n$. We define norms over $\mathcal{R}_q$ with respect to the coefficient embedding $\tau : \mathcal{R}_q \to \mathbb{Z}_q^d$, mapping $\boldsymbol{r}$ to the vector $(r_0, \ldots, r_{d-1})$. Thus, for $\boldsymbol{r} \in \mathcal{R}_q$, $\|\boldsymbol{r}\|_p := \|\tau(\boldsymbol{r})\|_p$. This can naturally be generalized to norms over $\mathcal{R}_q^n$ by extending the coefficient embedding to $\tau : \mathcal{R}_q^n \to \mathbb{Z}_q^{nd}, \vec{\boldsymbol{v}} \mapsto \tau(\boldsymbol{v}_1)\|\tau(\boldsymbol{v}_2)\| \ldots \|\tau(\boldsymbol{v}_n)$. Additionally, we define the operator norm of $\boldsymbol{r} \in \mathcal{R}_q$ as

$$\|\boldsymbol{r}\|_{\mathrm{op}} = \max_{\boldsymbol{s}\in\mathcal{R}_q} \frac{\|\boldsymbol{r}\boldsymbol{s}\|_2}{\|\boldsymbol{s}\|_2}.$$

We write $r \leftarrow \mathcal{D}$ to denote that r was sampled from the distribution $\mathcal{D}$. When sampling uniformly at random from a set S, we use the shorthand $r \xleftarrow{\$} S$. By $\lceil\cdot\rfloor$ we denote rounding up, which is extended to vectors in the natural way. For a positive integer n we let $[n] = \{1, \ldots, n\}$.

2.2 Power-of-Two Cyclotomic Rings

Let d be a power-of-two and $l \mid d$ such that $q = 2l + 1 \bmod 4l$. This condition ensures that $\mathbb{Z}_q$ contains a primitive $2l$-th root of unity ζ, but no element of order a larger power of 2. Then modulo q, $X^d + 1$ factors as the product of l irreducible polynomials $X^\delta - \zeta_i$ of degree $\delta = d/l$, where $\zeta_i = \zeta^{2i-1}$ for $i = 1, \ldots, l$ [LS18, Cor. 1.2]. By the Chinese remainder theorem (CRT), it follows that the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ splits into the product of l residue fields

$$\mathcal{R}_q \cong \mathbb{Z}_q[X]/\langle X^\delta - \zeta_1\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle X^\delta - \zeta_l\rangle.$$

We call l the *split factor* and δ the *split ratio* of $\mathcal{R}_q$.

For a given degree d, we can vary how to choose the modulus q in order to achieve different splitting behaviors, which in turn affect the mathematical properties of $\mathcal{R}_q$. There are mainly three splitting regimes studied in the literature. We call R_q *fully-splitting* if $l = d$ and thus $\delta = 1$. We call it *almost-fully-splitting* if $l = d/2^c$ and thus $\delta = 2^c$ for a small positive integer c. In this paper, we focus on $c \in \{2, 3\}$. Lastly, we say that R_q is *two-splitting* if $l = 2$ and thus $\delta = d/2$.

For an element $\boldsymbol{r} \in \mathcal{R}_q$, we call $\boldsymbol{r} \bmod (X^\delta - \zeta_i)$ the i-th CRT-slot of $\boldsymbol{r}$. An element is invertible in $\mathcal{R}_q$ if and only if all of its CRT-slots are non-zero. We denote by $\mathcal{R}_q^\times$ the set of invertible ring elements.

Power-of-two cyclotomic rings enjoy a special popularity in the design of cryptographic schemes as they come with nice properties. As detailed out in [LNPS21, Sec. 2.8], it is possible to nicely work over subrings of $\mathcal{R}$. More precisely, there is a norm-preserving bijection $\phi\colon \mathcal{R} \to \mathcal{S}^c$, where $\mathcal{R}$ is of degree d and $\mathcal{S}$ is of degree $d' = d/c$ for some positive integer c. The bijection can be naturally extended to vectors over $\mathcal{R}$ and $\mathcal{S}$ and the respective quotient rings modulo q.

The ring $\mathcal{R}_q$ inherits its group of automorphisms $\mathsf{Aut}(\mathcal{R}_q)$ from the Galois automorphisms of the $2l$-th cyclotomic number field [ALS20], i.e., $\mathsf{Aut}(\mathcal{R}_q) = \{\sigma_i \mid i \in \mathbb{Z}_{2l}^\times\} \cong \mathbb{Z}_{2l}^\times$, where σ_i is defined by $X \mapsto X^i$ and $\mathbb{Z}_{2l}^\times$ denotes the multiplicative group of unit of $\mathbb{Z}_{2l}$.

The conjugation automorphism σ_{-1} is of special interest to us in this paper. As observed in [LNP22, Lemma 2.4], for power-of-two cyclotomics, σ_{-1} relates inner products in $\mathcal{R}_q^n$ to the inner products of their coefficient vectors by

$$\langle \tau(\vec{\boldsymbol{a}}), \tau(\vec{\boldsymbol{b}})\rangle = \mathsf{ct}\left(\langle \sigma_{-1}(\vec{\boldsymbol{a}}), \vec{\boldsymbol{b}}\rangle\right) \text{ for } \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \in \mathcal{R}_q^n. \tag{1}$$

In particular, we have that $\|\vec{\boldsymbol{a}}\|_2^2 = \mathsf{ct}\left(\langle \sigma_{-1}(\vec{\boldsymbol{a}}), \vec{\boldsymbol{a}}\rangle\right)$. The j-th coefficient of some $\boldsymbol{a} = \sum_{i=1}^{d-1} a_i X^i \in \mathcal{R}_q$ can be retrieved through $\mathsf{ct}\left(\sigma_{-1}(X^j)\boldsymbol{a}\right) = a_j$, as multiplying by X^{-j} shifts the j-th coefficient to the first position.

Finally, we are going to need the following result.

Lemma 2.1 ([AL21, Prop. 2]). *The expansion factor of $\mathcal{R}$ is defined as $\gamma_{\mathcal{R}} \coloneqq \max_{\boldsymbol{a},\boldsymbol{b}\in\mathcal{R}} \frac{\|\boldsymbol{ab}\|_\infty}{\|\boldsymbol{a}\|_\infty \|\boldsymbol{b}\|_\infty}$. When d is a power of 2, $\gamma_{\mathcal{R}} \leq d$.*

2.3 Aggregate Signatures, Falcon, and SNARKs

In supplementary material B, we recall Falcon [PFH+22] and standard definitions for aggregate signatures (denoted by $\mathsf{AS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{AggSign}, \mathsf{AggVer})$, of which the first four define a usual signature scheme S), and (succinct) non-interactive arguments (denoted by $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$).

2.4 LaBRADOR Proof System

Before describing how we use and modify LaBRADOR [BS23] to aggregate Falcon signatures, we will for exposition purposes give a quite extensive summary of the protocol. The complete protocol description is provided in Appendix B.6.

Ring and Challenge Space. Their protocol is presented over a two-splitting $\mathcal{R}_q$. Thus, one can use [LS18, Corollary 1.2] to construct a challenge space $\mathcal{C}$ satisfying the following properties relevant for soundness and compact proof sizes: (1) It is exponentially large in the security parameter, i.e., $|\mathcal{C}| \geq 2^\lambda$. (2) Given two distinct challenges $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}$, their difference $\bar{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c}'$ is always invertible in $\mathcal{R}_q$. (3) The polynomials $\boldsymbol{c} \in \mathcal{C}$ have small norm. Let $T_2, T_{\mathrm{op}} \in \mathbb{R}$ be such that $\|\boldsymbol{c}\|_2^2 \leq T_2$ and $\|\boldsymbol{c}\|_{\mathrm{op}} \leq T_{\mathrm{op}}$ for all $\boldsymbol{c} \in \mathcal{C}$.

Relation. We now describe the relation R for which LaBRADOR has been designed. Informally, the relation R consists of a collection of dot product constraints and a norm bound check over $\mathcal{R}_q$. More formally, Let $\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r \in \mathcal{R}_q^n$ be r witness vectors of rank n. Each dot product constraint is defined by a function f of the form

$$f(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) = \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_i \rangle - \boldsymbol{b} \in \mathcal{R}_q,$$

where $\boldsymbol{a}_{i,j}, \boldsymbol{b} \in \mathcal{R}_q$ and $\vec{\boldsymbol{\varphi}} \in \mathcal{R}_q^n$, such that $\boldsymbol{a}_{i,j} = \boldsymbol{a}_{j,i}$ for all i, j. For some of these functions f', we are only interested in the constant term $\mathsf{ct}(\cdot)$ of their output. Therefore, we separate the full dot product constraints $\mathcal{F}$ from the constant term constraints $\mathcal{F}'$. Finally, we define the relation R such that $\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r$ is a witness for the statement $(\mathcal{F}, \mathcal{F}', \beta)$ if and only if

$$(\forall f \in \mathcal{F}, f(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) = \mathbf{0}) \wedge (\forall f' \in \mathcal{F}', \mathsf{ct}\left(f'(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r)\right) = 0) \wedge \sum_{i=1}^{r} \|\vec{\boldsymbol{w}}_i\|_2^2 \leq \beta^2.$$

Modular Johnson-Lindenstrauss Lemma. To prove that the witness vectors have short ℓ_2-norm, without sending the entire witness $\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r$ to the verifier, LaBRADOR uses the projection technique from [GHL22]. It is based on a modular version of the Johnson-Lindenstrauss lemma, which was introduced in [GHL22] and heuristically strengthened in [BS23]. Whereas [BS23] stated the result for the concrete security parameter $\lambda = 128$, we generalize it to arbitrary λ.

Lemma 2.2 (Adapted from [BS23], heuristic). *Let C be the distribution where 0 has probability $\frac{1}{2}$ and ± 1 both have probability $\frac{1}{4}$. Let $q \in \mathbb{N}$. For every $\lambda \in \mathbb{N}$, there exist constants $C_1 = C_1(\lambda)$ and $C_2 = C_2(\lambda)$ such that the following holds. For every vector $\vec{w} \in \mathbb{Z}_q^n$ with $\|\vec{w}\|_2 \geq b$ for some bound $b \leq q/C_1$,*

$$\Pr_{\Pi \leftarrow \mathsf{C}^{2\lambda \times n}} \left[\|\Pi \vec{w} \bmod q\|_2 < \sqrt{C_2} b \right] \lesssim 2^{-\lambda},$$

where $\lesssim$ stresses the heuristic nature of the result.

We provide a Python script jl.py, accessible in our repository, which computes the constants for different security levels λ. In this work, we are interested in two security levels. For $\lambda = 128$, we obtain $C_1 = 120$ and $C_2 = 30$.[9] For $\lambda = 256$, we obtain $C_2 = 168$ and $C_2 = 60$.

To prove that $\vec{\boldsymbol{w}} \in \mathcal{R}_q^n$ is short, the verifier sends the random projection matrix $\Pi \leftarrow \mathsf{C}^{2\lambda \times (nd)}$, and the prover responds with the projection $\vec{p} = \Pi \tau(\vec{\boldsymbol{w}}) \in \mathbb{Z}_q^{2\lambda}$. If $\|\vec{\boldsymbol{w}}\|_2 \leq \beta$, then the expected ℓ_2-norm of $\vec{p}$ is $\sqrt{\lambda}\beta$. The verifier checks whether $\|\vec{p}\|_2 \leq \sqrt{\lambda}\beta$, which happens in the honest case with probability 1/2. If this norm check holds and $\sqrt{\lambda/C_2}\beta < q/C_1$, then by the lemma, $\|\vec{\boldsymbol{w}}\|_2 \leq (\sqrt{\lambda/C_2})\beta$ with overwhelming probability. Hence, this is an approximate norm proof of constant size with *slack* $\sqrt{\lambda/C_2}$. For both $\lambda \in \{128, 256\}$, the slack equals $\sqrt{128/30} \approx 2$. Notice that in the non-interactive variant, the projection matrix Π can be generated from a short λ-bit seed.

3 Choice of Ring and Challenge Space

Let $\mathcal{R}_q = \prod_{i=1}^{l} \mathbb{Z}_q[X]/\langle X^\delta - \zeta_i \rangle$, where $d = l \cdot \delta$, with d the ring degree, l the split factor and δ the split ratio of $\mathcal{R}_q$, as introduced in Section 2.2.

As recalled in Section 2.4, LaBRADOR [BS23] uses the properties of their two-splitting ring (i.e., $l = 2$) to design a useful challenge space $\mathcal{C} \subset \mathcal{R}_q$ for their protocol. In particular, they make use of the facts that in two-splitting rings 1) every element in $\mathcal{C}$ is invertible and 2) the size of each of the two CRT-slots is exponentially big in the ring degree d. However, two-splitting rings do not allow for fast computations such as the number theoretic transform and thus lead to very slow computation times (cf. Section 6.2 for concrete numbers).

We propose an improvement of LaBRADOR by moving from the two-splitting to the setting of almost-fully-splitting rings (i.e., $\delta = 2^c$ for a small positive integer c) with only a small loss in proof sizes, and thus with only a small loss in aggregate signature sizes (cf. Section 6.1 for concrete numbers). In this case, we can only guarantee that an element of $\mathcal{C}$ is invertible with probability at least $1 - 2^{-\lambda}$ for security level λ. To address the technical challenges we faced when doing so, we introduce the property of well-spread challenge spaces in Section 3.1. Moreover, in Section 3.2, we generalize Schwartz-Zippel to almost-fully-splitting rings by connecting it to the previously introduced well-spreadness. We believe that both contributions might be of independent interest for the design of future lattice-based SNARKs. We explain in Appendix D.1 why we did not chosse the fully-splitting setting.

[9] Note that our constant C_1 for $\lambda = 128$ is slightly smaller than the one in [BS23], which was 125, and thus slightly tightens the result.

3.1 Well-Spread Challenge Spaces

In this section, we define the property of well-spreadness of a challenge set. Our hope is that it abstracts away the property we need from $\mathcal{C}$ and that it will be easier in the future to instantiate it with different challenge space constructions.

Definition 3.1. *Let $\mathcal{C} \subset \mathcal{R}_q$ and $B \in [0,1]$. We say that $\mathcal{C}$ is* B-well-spread *if for all $i \in [l]$ and for all $\boldsymbol{y} \in \mathbb{Z}_q[X]/\langle X^\delta - \zeta_i \rangle$*

$$\Pr_{\boldsymbol{c} \xleftarrow{\$} \mathcal{C}} [\boldsymbol{c} \bmod (X^\delta - \zeta_i) = \boldsymbol{y}] \leq B.$$

Informally, well-spreadness bounds the probability that any CRT-slot of a random challenge element hits a specific element. Implicitly, well-spreadness has already be shown for different challenge sets such as the one in [ALS20, Lem. 3.2], the one in [ESZ22, Lem. 1] and its generalization in [ESLR23, Lem. 1]. Well-spreadness directly implies a bound on invertibility of randomly sampled challenges (by setting $\boldsymbol{y} = \boldsymbol{0}$ in the lemma below) as well as the invertibility of challenge differences (by setting $\boldsymbol{y} = \boldsymbol{c}'$ for $\boldsymbol{c}' \in \mathcal{C}$). The proof is deferred to Appendix D.2.

Lemma 3.1. *Let $\mathcal{C} \subset \mathcal{R}_q$ be B-well-spread for $B \in [0,1]$. Let $\boldsymbol{y} \in \mathcal{R}$ by an arbitrary ring element. It yields*

$$\Pr_{\boldsymbol{c} \xleftarrow{\$} \mathcal{C}} [\boldsymbol{c} - \boldsymbol{y} \in \mathcal{R}_q^\times] \geq 1 - l \cdot B.$$

For our instantiation of LaBRADOR, we use the challenge space proposed in [ESZ22] and generalized in [ESLR23]. They already (implicitly) showed that the challenge space is well-spread and provided a slow exact as well as a fast heuristic MathSage script to compute the well-spreadness bound B for different parameter sets. We defer to Appendix D.4 for a detailed description of the concrete instantiation and parameter settings.

3.2 Variant of Schwartz-Zippel Lemma

Informally, the Schwartz-Zippel lemma bounds the probability that a polynomial evaluated on random challenge elements yields the zero element. However, the original lemma works over fields, as there we can relate the number of distinct roots of a polynomial to its degree. In our proof system, we need to apply Schwartz-Zippel over the ring $\mathcal{R}_q$, which is in general not a field. We thus propose the following generalization of Schwartz-Zippel to rings, as long as the challenge space is well-spread. The proof is deferred to Appendix D.3.

Lemma 3.2. *Let $n \in \mathbb{N}$ and $\boldsymbol{f} \in \mathcal{R}_q[X_1, \ldots, X_n]$ be a non-zero polynomial over $\mathcal{R}_q$ of total degree $\deg(\boldsymbol{f}) \geq 0$. Further, let $\mathcal{C} \subset \mathcal{R}_q$ be a B-well-spread challenge space for $B \in [0,1]$. Let $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ be sampled independently and uniformly at random from $\mathcal{C}$. Then*

$$\Pr[\boldsymbol{f}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) = \boldsymbol{0}] \leq \deg(\boldsymbol{f}) \cdot B.$$

Remark 3.1. Note that LaBRADOR [BS23] already (implicitly) use Schwartz-Zippel over the ring $\mathcal{R}_q$. However, they are making use of the specific structure of their two-splitting ring setting and their challenge space. In particular, by [LS18], *any* challenge difference $\boldsymbol{c} - \boldsymbol{c}'$ of LaBRADOR is invertible in $\mathcal{R}_q$, hence

$$\boldsymbol{c} - \boldsymbol{c}' \in \mathcal{R}_q^\times \Leftrightarrow \boldsymbol{c} - \boldsymbol{c}' \neq \boldsymbol{0} \bmod (X^\delta - \zeta_i)\ \forall i \in [l] \Leftrightarrow \boldsymbol{c} \neq \boldsymbol{c}' \bmod (X^\delta - \zeta_i)\ \forall i \in [l].$$

In other words, every two distinct challenges have different CRT-slots. On the one hand, this implies that for every $i \in [l]$, sampling uniformly at random from $\mathcal{C}$ and then reducing modulo $X^\delta - \zeta_i$ is the same as directly sampling uniform at random from $\mathcal{C} \bmod X^\delta - \zeta_i$. On the other hand, it implies that the size of $\mathcal{C} \bmod X^\delta - \zeta_i$ is the same as the size of $\mathcal{C}$ itself. One can thus deduce that in the two-splitting case

$$Pr[\boldsymbol{f}(\boldsymbol{c}) = \boldsymbol{0}] \leq \deg(\boldsymbol{f})/|\mathcal{C}|.$$

Alternatively, one could use the results of [BCPS18, Thm. 4.2] proving Schwartz-Zippel over rings in which every challenge difference is invertible.

4 Predicate Special Soundness (PSS)

In the following section we assume familiarity with special-soundness of multi-round protocols, and associated notions such as trees of accepting transcripts. We additionally assume familiarity with the Fiat-Shamir transform, which converts interactive proofs to non-interactive proofs in the random oracle model. Complete definitions of each of these notions can be found in Appendix B.

Many multi-round protocols follow a commit-and-open structure, where the prover begins the protocol by committing to its witness, and concludes with some (possibly masked) opening of its witness. In the intermediate rounds additional properties of the witness are guaranteed by the use of probabilistic tests. For a fixed invalid witness the probability of passing these tests must be small. These extra guarantees go beyond what may be obtained by special-soundness.

The natural extraction strategy in the interactive setting, is to start by finding a tree which allows extracting a candidate witness [Ngu22]. For this first tree, the extracted witness may depend on the challenges, preventing the use of the bounds for the probabilistic tests. Instead, the extractor must find a second tree for the same commitment. If this second tree is for the same witness, then the independence of the freshly sampled challenges allows applying the bounds for the probabilistic tests. Alternatively, if the second tree results in a distinct opening, binding is broken, allowing a reduction to the security of the commitment scheme.

We systematize the equivalent approach in the non-interactive setting, extracting subtrees and enforcing properties on them by extracting an additional subtree. To describe the validity of extracted trees we introduce predicates which capture the additional guarantees obtained through probabilistic tests. Further, we also enable specialized predicates to obtain challenges with particular properties beyond just being distinct. For a broad class of predicates extraction techniques for special-sound protocols [AFK22] may be generalized to allow extracting valid trees, while keeping knowledge error small.

4.1 PSS Framework

Let us begin by introducing some notation for trees of transcripts. See also Figure 1 for graphical representation of each subtree.

Definition 4.1 (Tree of Transcripts). *Let $\mu, k_1, \ldots, k_\mu \in \mathbb{N}$ and let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $2\mu + 1$-message public-coin argument of knowledge for a relation R_{pp}. Additionally, let $m \in [\mu]$ and $\ell \in [k_m]$. Let $\mathcal{C}_m$ be the m-th challenge set.*

- *We define $\mathbb{T}_{\mu+1}$ be the set of possible accepting transcripts for Π.*
- *We define $\mathbb{T}^{(\ell)}_{m+1}$ be the set of possible accepting $(1, \ldots, 1, \ell, k_{m+1}, \ldots, k_\mu)$-trees of transcripts for Π, and denote $\mathbb{T}_m = \mathbb{T}^{(k_m)}_{m+1}$. Each $t \in \mathbb{T}^{(\ell)}_{m+1}$ is a tuple $t = (t_1, \ldots, t_\ell) \in \mathbb{T}^{\ell}_{m+1}$.*
- *For $t \in \mathbb{T}_{m+1}$, we define $\mathsf{trunk}(t)$ to be the prefix $(a_1, c_1, a_2, c_2, \ldots, a_m)$ shared by all the transcripts in t, and $\mathsf{chal}_i(t)$ for $i \in [m]$ to be the i-th round challenge $c_i \in \mathcal{C}_i$ shared by the transcripts.*
- *Let $\mathcal{C}^{(\ell)}_m$ be the set of $(c_1, \ldots, c_\ell) \in \mathcal{C}_m$ with $c_{i_1} \neq c_{i_2}$ for all $i_1 \neq i_2$. These are all the combinations of m-th challenges that may occur in $\mathbb{T}^{(\ell)}_{m+1}$.*

Remark 4.1. We assume that the first message of each transcript includes the statement. Thus, any function taking a tree as an argument is also implicitly given the statement and any transcripts with the same first message share the same statement. This also allows us to directly handle the case of adaptive knowledge soundness.

Each predicate is associated with a level in the $(k_1, \ldots, k_\mu)$-tree of transcripts. In practice a predicate relates to the probabilistic test for the challenge in associated round of the protocol. We divide predicates into two types, challenge predicates and commitment predicates.

Challenge predicates allow enforcing relationships between challenges beyond them simply being distinct. Level m has k_m challenge predicates, where the ℓth predicate takes the first ℓ challenges as input. When checked in sequence each predicate may therefore be seen as ensuring guarantees for one new challenge with respect the set of challenges found earlier. This allows capturing the case where two independently sampled challenges only allow extracting a witness with high probability.

Commitment predicates allow enforcing probabilistic tests. These tests are usually defined such that any fixed witness only allows a prover to cheat with a given probability p for an independently chosen challenge or, alternatively, any fixed witness permits a set of "bad" challenges of bounded size.

Let us consider a level which is $(2, k_2, \ldots, k_\mu)$-special-sound, where each subtree allows computing a witness which was previously committed to. Given just one of these two $(1, k_2, \ldots, k_\mu)$-subtrees the extracted witness may depend on the challenge used for the probabilistic test. However, if the first challenge in the second $(1, k_2, \ldots, k_\mu)$-subtree is sampled independently, then for the same witness there is a low probability the challenge is in the "bad" set. This allows the guarantees of the test to be obtained with probability close to $1 - p$. In the case where the witness found for the second subtree is distinct we have instead found a violation of binding and no longer need to ensure our desired property.

In our framework we present a slight generalization of this approach, allowing a fork with k subtrees. On an intuitive level the first $k - 1$ subtrees allow extracting an opening and the final subtree must be consistent with that opening, breaking binding if it is inconsistent. Note, while we do not obtain two complete openings with this approach a single subtree may provide a partial opening which for some commitment schemes is sufficient to break binding. The requirements to break binding are described through a binding predicate, while the property we wish to ensure is captured by a property predicate. We call a pair of a binding and property predicate a commitment predicate.

Definition 4.2 (Predicates). *Let $m \in [\mu]$ and $\ell \in [k_m]$.*

1. *A challenge predicate on level m for the ℓth challenge is a function*

$$\Phi^{\mathsf{chal}}_{m,\ell} : \mathcal{C}^{(\ell)}_m \to \{0, 1\}.$$

2. *A commitment predicate on level m is a pair $\left(\Phi^{\mathsf{prop}}_m, \Phi^{\mathsf{bind}}_m\right)$, where*

$$\Phi^{\mathsf{prop}}_m : \mathbb{T}^{(k_m-1)}_{m+1} \to \{0, 1\} \quad \textit{and} \quad \Phi^{\mathsf{bind}}_m : \mathbb{T}^{(k_m-1)}_{m+1} \times \mathbb{T}_{m+1} \to \{0, 1\}.$$

To express whether a complete tree of transcripts is valid we collect our predicates to form a predicate system. In a predicate system the validity of a tree is defined recursively starting from single accepting transcripts which we always consider to be valid. Predicates are then enforced bottom-up, first grouping transcripts which fork only in the last challenge to form a new $(1, \ldots, 1, k_\mu)$-subtree. Such a subtree is valid if the predicates for level μ are satisfied.

Subsequently, $k_{\mu-1}$ of these subtrees are then grouped again to form a $(1, \ldots, k_{\mu-1}, k_\mu)$-subtree. Validity now follows if each $(1, \ldots, 1, k_\mu)$-subtree is valid and the predicates for level $\mu - 1$ hold. This procedure is iterated until the root of the tree is reached, defining validity for a complete $(k_1, \ldots, k_\mu)$-tree. We describe validity formally in terms of a boolean function over trees below.

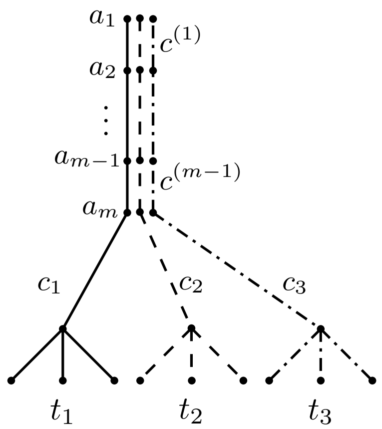
Definition 4.3 (Predicate System). *A predicate system $\boldsymbol{\Phi}$ for a $(k_1, \ldots, k_\mu)$-tree structure is a collection of predicates for each level in the tree. The m-th level has one commitment predicate $\left(\Phi^{\mathsf{prop}}_m, \Phi^{\mathsf{bind}}_m\right)$, and k_m challenge predicates $\Phi^{\mathsf{chal}}_{m,1}, \ldots, \Phi^{\mathsf{chal}}_{m,k_m}$. We recursively define a series of boolean function $\boldsymbol{\Phi}_m$ for $m \in [\mu + 1]$, describing whether a partial tree of transcripts satisfies the predicate system. For a single accepting transcript $t \in \mathbb{T}_{\mu+1}$ we let $\boldsymbol{\Phi}_{\mu+1}(t) = 1$. For all larger subtrees $t = (t_1, \ldots, t_{k_m}) \in \mathbb{T}_m$ for some $m \in [\mu]$ with $\boldsymbol{c}_i = \mathsf{chal}_m(t_i)$, then $\boldsymbol{\Phi}_m(t) = 1$ if and only if*

$$\begin{aligned}\left(\bigwedge_{i \in [k_m]} \boldsymbol{\Phi}_{m+1}(t_i) = 1 \wedge \Phi^{\mathsf{chal}}_{m,i}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_i) = 1\right) &\wedge \Phi^{\mathsf{prop}}_m(t_1, \ldots, t_{k_m-1}) = 1 \\ &\wedge \Phi^{\mathsf{bind}}_m((t_1, \ldots, t_{k_m-1}), t_{k_m}) = 1.\end{aligned}$$

For notational convenience, we let $\boldsymbol{\Phi} = \boldsymbol{\Phi}_1$.

Having defined what it means for a tree to satisfy a predicate system we may now proceed to define predicate special soundness. We extend the notion of special-soundness to additionally require a predicate system to be valid for successful extraction to be guaranteed. Note, predicate special soundness is equivalent to special-soundness if the predicate system only contains predicates which are trivially true.

Definition 4.4 (Predicate Special Soundness). *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $2\mu + 1$-message public-coin argument of knowledge for a relation R_{pp}. We say that Π is $(\boldsymbol{K}, \boldsymbol{\Phi})$-predicate-special-sound for $\boldsymbol{K} = (k_1, \ldots, k_\mu)$ and a predicate system $\boldsymbol{\Phi}$ if there exists a polynomial time algorithm which given a statement x and a $\boldsymbol{K}$-tree of transcripts t for this statement with $\boldsymbol{\Phi}(t) = 1$ always outputs a witness w such that $(x, w) \in R_{\mathsf{pp}}$.*



$$t = (t_1, t_2, t_3),\ \mathsf{chal}_m(t_1) = c_1,$$
$$\mathsf{trunk}(t) = (a_1, c^{(1)}, \dots, a_m),$$
$$\mathbf{\Phi}(t) = 1 \Leftrightarrow \begin{cases} \mathbf{\Phi}(t_1) = \mathbf{\Phi}(t_2) = \mathbf{\Phi}(t_3) = 1 \\ \Phi^{\mathsf{chal}}_{m,1}(c_1) = 1 \\ \Phi^{\mathsf{chal}}_{m,2}(c_1, c_2) = 1 \\ \Phi^{\mathsf{chal}}_{m,3}(c_1, c_2, c_3) = 1 \\ \Phi^{\mathsf{prop}}_{m}(t_1, t_2) = 1 \end{cases}$$

Fig. 1: Constructing $t \in \mathbb{T}_m$ from trees $t_1, t_2, t_3 \in \mathbb{T}_{m+1}$ with a common trunk and $k_m = 3$ distinct level m challenges. Here $(t_1, t_2) \in \mathbb{T}^{(2)}_{m+1}$.

Not all predicate systems allow for efficient extraction. We describe one large natural class of predicates that do allow efficient extraction, characterizing them in terms of their *failure density*. The failure density of our predicates directly contributes to the knowledge error of the protocol.

For the ℓth challenge predicate on a level m, the failure density of $\Phi^{\mathsf{chal}}_{m,\ell}$ bounds the proportion of new challenges which do not satisfy the predicate for any fixed set of challenges $\boldsymbol{c}_1, \dots, \boldsymbol{c}_{\ell-1}$. For extraction to succeed, each challenge predicate in the tree must be satisfied. Thus, the failure density of challenge predicates allows bounding the probability that each new subtree the extractor finds causes failure.

Definition 4.5 (Failure Density of Challenge Predicates). *Let $m \in [\mu]$ and $\ell \in [k_m]$. Let $c_1, \dots, c_{\ell-1} \in \mathcal{C}_m$ be any $\ell - 1$ distinct challenges such that $\Phi^{\mathsf{chal}}_{m,i}(c_1, \dots, c_i) = 1$ for all $i \in [\ell - 1]$. Consider the set of possible ℓ-th challenges such that $\Phi^{\mathsf{chal}}_{m,\ell}$ fails,*

$$B(c_1, \dots, c_{\ell-1}) = \left\{ c \in \mathcal{C}_m \setminus \{c_1, \dots, c_{\ell-1}\} \;\middle|\; \Phi^{\mathsf{chal}}_{m,\ell}(c_1, \dots, c_{\ell-1}, c) = 0 \right\}.$$

The challenge predicate $\Phi^{\mathsf{chal}}_{m,\ell}$ has failure density $p^{\mathsf{chal}}_{m,\ell}$ if it always holds that $|B(c_1, \dots, c_{\ell-1})| \le p^{\mathsf{chal}}_{m,\ell} |\mathcal{C}_m|$.

For any fixed set of valid subtrees $t_1, \dots, t_{k_m-1}$, which do not satisfy the new desired property on a level m, the failure density of a commitment predicate $\Phi^{\mathsf{com}}_m = (\Phi^{\mathsf{prop}}_m, \Phi^{\mathsf{bind}}_m)$ bounds the fraction of challenges which would allow an additional accepting subtree t_{k_m} without violating binding.

Definition 4.6 (Failure Density of Commitment Predicates). *Let $m \in [\mu]$. Define a set of bad subtrees*

$$\mathsf{Bad}^{\mathsf{prop}}_m = \left\{ (t_1, \dots, t_{k_m-1}) \in \mathbb{T}^{(k_m-1)}_{m+1} \;\middle|\; \begin{array}{c} \forall i \in [k_m - 1]:\ \boldsymbol{\Phi}_{m+1}(t_i) = 1, \\ \Phi^{\mathsf{chal}}_{m,i}(c_1, \dots, c_i) = 1, \\ \Phi^{\mathsf{prop}}_m(t_1, \dots, t_{k_m-1}) = 0 \end{array} \right\}$$

using the shorthand $c_i = \mathsf{chal}_m(t_i)$. That is, subtrees in $\mathsf{Bad}^{\mathsf{prop}}_m$ fail to satisfy the property predicate but otherwise satisfy the constraints. For each $(t_1, \dots, t_{k_m-1}) \in \mathbb{T}^{(k_m-1)}_{m+1}$, let $\mathsf{Sat}^{\mathsf{bind}}_m(t_1, \dots, t_{k_m-1})$ be the set of possible k_m-th subtrees that satisfy the binding predicate and the other constraints.

$$\mathsf{Sat}^{\mathsf{bind}}_m(t_1, \dots, t_{k_m-1}) = \left\{ t \in \mathbb{T}_{m+1} \;\middle|\; \begin{array}{c} (t_1, \dots, t_{k_m-1}, t) \in \mathbb{T}_m, \boldsymbol{\Phi}_{m+1}(t) = 1, \\ \Phi^{\mathsf{chal}}_{m,k_m}(c_1, \dots, c_{k_m-1}, \mathsf{chal}_m(t)) = 1, \\ \Phi^{\mathsf{bind}}_m((t_1, \dots, t_{k_m-1}), t) = 1 \end{array} \right\}$$

Consider the m-th level challenges occuring for some tree in this set,

$$B(t_1, \dots, t_{k_m-1}) = \{ \mathsf{chal}_m(t) \;\mid\; t \in \mathsf{Sat}^{\mathsf{bind}}_m(t_1, \dots, t_{k_m-1}) \}.$$

The commitment predicate $\left(\Phi^{\mathsf{prop}}_m, \Phi^{\mathsf{bind}}_m\right)$ has failure density p^{com}_m if $|B(t_1, \dots, t_{k_m-1})| \le p^{\mathsf{com}}_m |\mathcal{C}_m|$ for all $(t_1, \dots, t_{k_m-1}) \in \mathsf{Bad}^{\mathsf{prop}}_m$.

When defining our knowledge extractor we wish to either extract a valid witness for our original relation, or find a solution for the computational problem the security of our commitment scheme depends on, we let such a solution be a witness for a new binding relation. Thus, if the problem is indeed hard, the proof system is sound. We only bound the failure density of our commitment predicates when the binding predicate is satisfied; we must be able to extract a witness for the binding relation when this is not the case, otherwise we would neither find a valid witness nor break binding. More formally.

Definition 4.7 (Binding Relation). *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $(\boldsymbol{K}, \boldsymbol{\Phi})$-predicate-special-sound argument of knowledge for a relation R_{pp}, and let R_{bind} be an additional relation. We say that $\boldsymbol{\Phi}$ admits $R_{\mathsf{bind,pp}}$ as a binding relation for public parameters* pp *if there exists a polynomial time algorithm $\mathcal{B}$, which on input $(t_1, \ldots, t_{k_m}) \in \mathbb{T}_m$ for some $m \in [\mu]$ such that $\Phi_m^{\mathsf{bind}}((t_1, \ldots, t_{k_m-1}), t_{k_m}) = 0$ and*

$$\forall i \in [k_m] : \ \boldsymbol{\Phi}_{m+1}(t_i) = 1 \text{ and } \Phi_{m,i}^{\mathsf{chal}}(\mathsf{chal}_m(t_1), \ldots, \mathsf{chal}_m(t_i)) = 1,$$

always satisfies $\mathcal{B}(t_1, \ldots, t_{k_m}) \in R_{\mathsf{bind,pp}}$.

We may now finally describe the knowledge soundness of a predicate-special-sound protocol.

Theorem 4.1. *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $(\boldsymbol{K}, \boldsymbol{\Phi})$-predicate-special-sound argument of knowledge for a relation R_{pp}. In addition, let $R_{\mathsf{bind,pp}}$ be a binding relation for $\boldsymbol{\Phi}$ with statement x. Then the adaptive Fiat-Shamir transformation $FS[\Pi]$ is adaptively knowledge sound for the relation $R_{\mathsf{pp}} \cup R_{\mathsf{bind,pp}}$ with knowledge error*

$$2(Q+1) \sum_{i=1}^{\mu} \max \left(\frac{k_i - 1}{|\mathcal{C}_i|}, p_i^{\mathsf{com}} + \sum_{\ell=1}^{k_i} p_{i,\ell}^{\mathsf{chal}} \right),$$

where Q is the number of random oracle queries made by the prover. The number of times that the knowledge extractor invokes the prover is in expectation at most $K + Q(K-1)$, where $K = \prod_{i=1}^{\mu} k_i$.

We prove this theorem in Section H. The achieved result is actually slightly stronger, in many cases allowing a smaller constant than 2.

4.2 Applying PSS

We now present an example of how PSS can be applied to a protocol. Protocol 1 is a commit and open protocol, proving knowledge of a short (strong) opening. Note that Protocol 1 can be viewed as a bare-bones version of LaBRADOR without aggregation and amortization. We extract a witness for the relation

$$R_{\mathsf{pp}} = \{(\vec{\boldsymbol{u}}, \vec{\boldsymbol{s}}) \mid \boldsymbol{A} \cdot \vec{\boldsymbol{s}} = \vec{\boldsymbol{u}}, \ \|\vec{\boldsymbol{s}}\|_2 \leq \sqrt{\lambda/C_2}\beta'\},$$

for the appropriate norm bound β', security parameter λ and a constant C_2.

The prover is given a witness $\vec{\boldsymbol{s}} \in \mathcal{R}_q^n$ and sends the commitment $\vec{\boldsymbol{u}} = \boldsymbol{A} \cdot \vec{\boldsymbol{s}}$ to the verifier, using a public matrix $\boldsymbol{A} \in \mathcal{R}_q^{\kappa \times n}$. The verifier then sends Johnson-Lindenstrauss projections $(\vec{\boldsymbol{\pi}}_j)_j$, which the prover applies to its witness, sending the resulting $(\boldsymbol{h}_j)_j$. Recall that τ is the coefficient embedding. Finally, the verifier sends a challenge $\boldsymbol{c}$ and the prover responds with the corresponding opening $\vec{\boldsymbol{z}}$. This last step is somewhat artificial, as we neither have zero-knowledge nor multiple witnesses, but allows illustrating the techniques needed for both cases.

Predicate Special Soundness First we show that Protocol 1 is $((2,2), \boldsymbol{\Phi})$-predicate-special-sound, for a predicate system $\boldsymbol{\Phi}$. On a high level, we wish to extract a weak opening and correct projection from $(1,2)$-trees, and then strengthen this to a strong opening when combining these to form a $(2,2)$-tree. A vector $\vec{\boldsymbol{s}}$ together with a challenge difference $\bar{\boldsymbol{c}}$ is a weak opening to a commitment $\vec{\boldsymbol{u}}$ if $\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{u}}$ and $\|\bar{\boldsymbol{c}}\vec{\boldsymbol{s}}\|_2 \leq \beta^*$ [ALS20]. The prover is bound to $\vec{\boldsymbol{s}}$ if Module-SIS is sufficiently hard for rank κ. While proving knowledge-soundness we show how two distinct weak openings allow computing a Module-SIS solution. We let all predicates which are not explicitly defined be trivial.

Level 2 Working bottom up, we may start by defining $\Phi_{2,2}^{\mathsf{chal}}(\boldsymbol{c}_1, \boldsymbol{c}_2) = 1 \Leftrightarrow \bar{\boldsymbol{c}} = (\boldsymbol{c}_1 - \boldsymbol{c}_2) \in \mathcal{R}_q^{\times}$, where $\boldsymbol{c}_1, \boldsymbol{c}_2$ come from the two transcripts t_1, t_2 such that $(t_1, t_2) \in \mathbb{T}_2$ we wish to combine. This challenge predicate is all we need to extract our desired opening. To disambiguate variables between two transcripts t_1 and t_2, we refer to them using the same subscript as their transcript. At this point we may assume $\boldsymbol{\Phi}_3(t_i) = 1$, i.e. t_1 and t_2 are accepting transcripts. It follows,

$$\boldsymbol{A}(\vec{\boldsymbol{z}}_1 - \vec{\boldsymbol{z}}_2) = (\boldsymbol{c}_1 - \boldsymbol{c}_2)\vec{\boldsymbol{u}}.$$

Protocol 1: 5-round Toy Example

$\mathcal{P}(\vec{s})$:		$\mathcal{V}()$:
$\vec{\boldsymbol{u}} = \boldsymbol{A} \cdot \vec{\boldsymbol{s}}$	$\xrightarrow{\vec{\boldsymbol{u}}}$	$\forall j \in [2\lambda] : \vec{\pi}_j \leftarrow \mathsf{C}^{nd}$
	$\xleftarrow{(\vec{\boldsymbol{\pi}}_j)_{j\in[2\lambda]}}$	$\vec{\boldsymbol{\pi}}_j = \tau^{-1}(\vec{\pi}_j) \in \mathcal{R}_q^n$
$\boldsymbol{h}_j = \langle \vec{\boldsymbol{\pi}}_j, \vec{\boldsymbol{s}} \rangle$	$\xrightarrow{(\boldsymbol{h}_j)_{j\in[2\lambda]}}$	$p_j = \mathsf{ct}(\boldsymbol{h}_j);\ \vec{p} = (p_1, \ldots, p_{2\lambda})$
	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \leftarrow \mathcal{C}_2$
$\vec{\boldsymbol{z}} = \boldsymbol{c} \cdot \vec{\boldsymbol{s}}$	$\xrightarrow{\vec{\boldsymbol{z}}}$	$\boldsymbol{A}\vec{\boldsymbol{z}} \stackrel{?}{=} \boldsymbol{c} \cdot \vec{\boldsymbol{u}}, \|\vec{\boldsymbol{z}}\|_2 \stackrel{?}{\leq} \beta,\ \|\vec{p}\|_2 \stackrel{?}{\leq} \sqrt{\lambda}\beta',$
		$\forall j \in [2\lambda] : \boldsymbol{c} \cdot \boldsymbol{h}_j \stackrel{?}{=} \langle \vec{\boldsymbol{\pi}}_j, \vec{\boldsymbol{z}} \rangle$

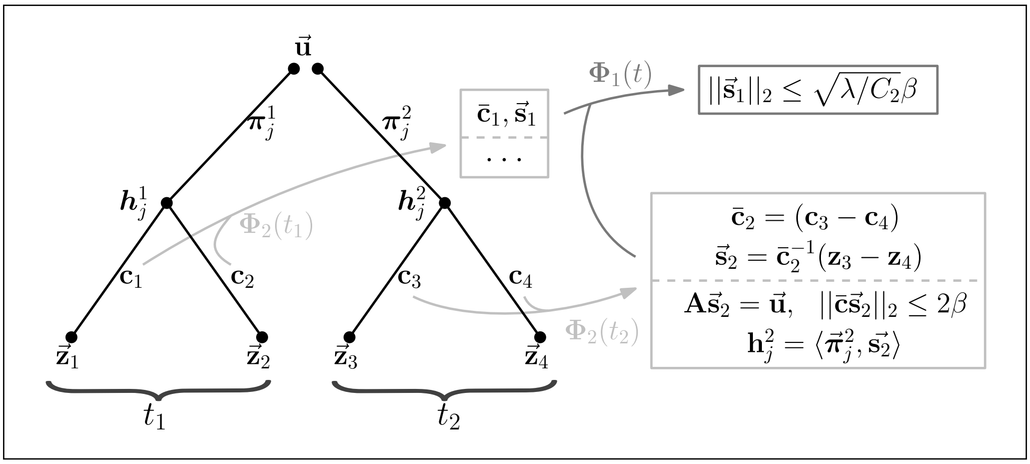


Fig. 2: Our extraction strategy for Protocol 1.

If $\Phi_{2,2}^{\mathsf{chal}}(\boldsymbol{c}_1, \boldsymbol{c}_2) = 1$ then $\vec{\boldsymbol{s}} = \bar{\boldsymbol{c}}^{-1}(\vec{\boldsymbol{z}}_1 - \vec{\boldsymbol{z}}_2)$ constitutes a weak opening, where $\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{u}}$ and $\|\bar{\boldsymbol{c}}\vec{\boldsymbol{s}}\|_2 \leq 2\beta$. Furthermore, for $j \in [2\lambda]$ we have $\bar{\boldsymbol{c}} \cdot \boldsymbol{h}_j = \langle \vec{\boldsymbol{\pi}}_j, \vec{\boldsymbol{z}}_1 - \vec{\boldsymbol{z}}_2 \rangle$. Again, if $\Phi_{2,2}^{\mathsf{chal}}(\boldsymbol{c}_1, \boldsymbol{c}_2) = 1$ then $\boldsymbol{h}_j = \langle \vec{\boldsymbol{\pi}}_j, \vec{\boldsymbol{s}} \rangle$ which implies that the projection was correctly computed, $\mathsf{ct}(\langle \vec{\boldsymbol{\pi}}_j, \vec{\boldsymbol{s}} \rangle) = p_j$.

On this level, the only non-trivial predicate is $\Phi_{2,2}^{\mathsf{chal}}$. If $\mathcal{C}_2$ is B-well-spread (cf. Definition 3.1) and $\mathcal{R}_q$ has a splitting factor l then the bound

$$\left|\left\{\boldsymbol{c} \in \mathcal{C}_2 \;\middle|\; \boldsymbol{c} \neq \boldsymbol{c}_1,\ \Phi_{2,2}^{\mathsf{chal}}(\boldsymbol{c}_1, \boldsymbol{c}) = 0\right\}\right| \leq l \cdot B \cdot |\mathcal{C}_2|,$$

follows by Lem. 3.1 for any fixed choice of $\boldsymbol{c}_1$, implying a failure density $p_{2,2}^{\mathsf{chal}} = l \cdot B$.

Level 1 We have now established that for a $(1,2)$-tree t where $\boldsymbol{\Phi}_2(t) = 1$ we may extract a weak opening, where the projection is correctly computed. However, we cannot be sure that the opening we have extracted is independent of the projection, meaning that $\vec{\boldsymbol{s}}$ might still be large. If $\vec{\boldsymbol{s}}$ is in fact large, then there is only a small fraction of the challenges that allow the prover to cheat, without breaking binding. Thus, we introduce a commitment predicate, by defining, a property predicate and a binding predicate. The property predicate describes the property of the witness we wish to enforce, in this case that the witness is short, i.e.

$$\Phi_1^{\mathsf{prop}}(t_1) = 1 \Leftrightarrow \|\vec{\boldsymbol{s}}_1\|_2 \leq \sqrt{\lambda/C_2}\beta'.$$

The binding predicate describes which values the prover is committed to between transcripts. If this predicate is violated we must be able to extract a solution to some hard problem.

$$\Phi_1^{\mathsf{bind}}(t_1, t_2) = 1 \Leftrightarrow \vec{\boldsymbol{s}}_1 = \vec{\boldsymbol{s}}_2.$$

Binding relation. We may show that $\boldsymbol{\Phi}$ admits the binding relation

$$R_{\mathsf{pp}}^{\mathsf{M\text{-}SIS}} = \{\ \vec{\boldsymbol{v}} \mid \vec{\boldsymbol{v}} \in \mathcal{R}_q^n,\ \boldsymbol{A} \cdot \vec{\boldsymbol{v}} = \vec{\boldsymbol{0}},\ 0 < \|\vec{\boldsymbol{v}}\|_2 \leq \beta^* \},$$

for a norm bound β^*. Consider trees t_1, t_2 such that $(t_1, t_2) \in \mathbb{T}_1$, where $\boldsymbol{\Phi}_2(t_1) = \boldsymbol{\Phi}_2(t_2) = 1$. As previously we distinguish variables in the two subtrees by using indices corresponding to their tree. We must show that a suitable witness may be extracted whenever $\Phi_1^{\mathsf{bind}}(t_1, t_2) = 0$, this happens exactly when $\vec{\boldsymbol{s}}_1 \neq \vec{\boldsymbol{s}}_2$. As they are both weak openings to the same commitment, $\boldsymbol{A}(\vec{\boldsymbol{s}}_1 - \vec{\boldsymbol{s}}_2) = \vec{\boldsymbol{0}}$, which in turn implies $\boldsymbol{A}\bar{\boldsymbol{c}}_1\bar{\boldsymbol{c}}_2(\vec{\boldsymbol{s}}_1 - \vec{\boldsymbol{s}}_2) = \vec{\boldsymbol{0}}$. All that remains is to argue that $\vec{\boldsymbol{v}} = \bar{\boldsymbol{c}}_1\bar{\boldsymbol{c}}_2(\vec{\boldsymbol{s}}_1 - \vec{\boldsymbol{s}}_2)$ is short. Using the triangle inequality and the definition of the operator norm we obtain, $\|\vec{\boldsymbol{v}}\| \leq \|\bar{\boldsymbol{c}}_2\|_{\mathrm{op}} \|\bar{\boldsymbol{c}}_1\vec{\boldsymbol{s}}_1\|_2 + \|\bar{\boldsymbol{c}}_1\|_{\mathrm{op}} \|\bar{\boldsymbol{c}}_2\vec{\boldsymbol{s}}_2\|_2$. Recall, we are working with weak openings where $\|\bar{\boldsymbol{c}}_i\vec{\boldsymbol{s}}_i\|_2 \leq 2\beta$. As $\bar{\boldsymbol{c}}_i$ is the difference of two challenges in $\mathcal{C}$ we also have $\|\bar{\boldsymbol{c}}\|_{\mathrm{op}} \leq 2T_{\mathrm{op}}$. Hence, $\vec{\boldsymbol{v}} \in R^{\mathsf{M\text{-}SIS}}_{\kappa,n,8T_{\mathrm{op}}\beta,\boldsymbol{A}}$.

Failure density. Having found an appropriate binding relation we may now turn to bounding failure density of a commitment predicate. Recall we wish to ensure the property $\|\vec{\boldsymbol{s}}\|_2 \leq \sqrt{\lambda/C_2}\beta'$. Fix some $t_1 \in \mathbb{T}_2$ such that $\boldsymbol{\Phi}_2(t_1) = 1$ and $\Phi_1^{\mathsf{prop}}(t_1) = 0$. Now we must consider the set

$$\mathsf{Sat}_m^{\mathsf{bind}}(t_1) = \left\{ t \in \mathbb{T}_2 \middle| \, (t_1, t) \in \mathbb{T}_1, \ \boldsymbol{\Phi}_2(t) = 1, \ \Phi_1^{\mathsf{bind}}(t_1, t) = 1 \right\}.$$

Due to $\boldsymbol{\Phi}_2(t) = 1$, we know that the projection must be correctly computed. Furthermore, Φ_1^{bind} ensures that all elements of $\mathsf{Sat}_m^{\mathsf{bind}}(t_1)$ are for the same value of $\vec{\boldsymbol{s}}$. We must bound the set of bad challenges, $B(t_1) = \{\mathsf{chal}_m(t) \ \mid \ t \in \mathsf{Sat}_m^{\mathsf{bind}}(t_1)\}$. By Lemma 2.2, for any fixed witness larger than $\sqrt{\lambda/C_2}\beta'$ the correctly computed projection is only smaller than $\sqrt{\lambda}\beta'$ with probability at most $p_1^{\mathsf{com}} = \mathsf{negl}(\lambda)$, giving, $|B(t_1)| \leq p_1^{\mathsf{com}}|\mathcal{C}_1|$.

Remark 4.2. Projection challenges, as expressed in Lemma 2.2, are not actually sampled uniformly. To match the uniform sampling required by the extraction framework one might instead consider a challenge set containing all bit strings of some fixed length. These strings may then be given as input to an algorithm sampling the desired distribution. In this case the distribution may be sampled perfectly giving the same bounds for the fraction of bad challenges. Note, in general, distinct bit strings may not necessarily give distinct challenges.

Knowledge Soundness By applying Theorem 4.1 we may conclude that the Fiat-Shamir transformation of Protocol 1 is knowledge sound for the relation $R_{\mathsf{pp}} \cup R_{\mathsf{pp}}^{\mathsf{M\text{-}SIS}}$ with knowledge error

$$2(Q+1)(\max\{1/|\mathcal{C}_1|, \ p_1^{\mathsf{com}}\} + \max\{1/|\mathcal{C}_2|, \ l \cdot B\}).$$

where the extractor invokes the prover at most $4 + 3Q$ times in expectation, and the prover makes at most Q oracle queries.

4.3 Extending to Coordinate-Wise PSS

To achieve sublinear proofs, modern lattice-based proof systems often favour a single amortized opening for a series of r witness, over a separate opening for each witness element. In recent work Fenzi *et al.* [FMN23] showed how these amortization techniques may be reconciled with extraction techniques designed for special-sound protocols. They achieved this by viewing protocols as being special-sound in r coordinates, allowing targeted extraction for each individual witness, resulting in the new notion of coordinate-wise special-soundness.

We show how a similar approach may be taken, extending PSS to coordinate-wise PSS. Importantly, this provides the final tool necessary for proving non-interactive LaBRADOR secure. For further details, see Appendix G.

4.4 Knowledge Soundness of LaBRADOR under the Fiat-Shamir Transform

We apply our PSS framework to prove that the Fiat-Shamir transformation of LaBRADOR is knowledge sound. For a full analysis we refer the reader to Appendix I. Here $R_{\sigma,\mathsf{pp}}$ is the LaBRADOR relation modified to allow a factor σ norm slack, formally (17).

Theorem 4.2. *Let Π be the base LaBRADOR protocol as described in Protocol 2 and 3. We consider the case with a ring $\mathcal{R}_q$ of degree d with splitting factor l and a B-well-spread challenge set, where each challenge has operator norm at most T_{op}. Restrict statements in the relation to have ℓ_2 norm bound β at most q/C_1, where C_1, C_2 are the parameters of Lemma 2.2. Let $\mathsf{adv}_{\mathsf{M\text{-}SIS},n,\beta}$ be an upper bound on the reduction's advantage in solving the* M-SIS *problem with module rank n and norm β. Then the*

Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ *is adaptively knowledge sound for the relation* $R_{\sigma,\mathsf{pp}}$ *where* $\sigma = \sqrt{\lambda/C_2}$ *with knowledge error*

$$2(Q+1)\left(2^{-\lambda} + (5+2l)Br + q^{-d/l} + q^{-\lceil \lambda/\log_2 q \rceil}\right) \tag{2}$$

$$+ \mathsf{adv}_{\mathsf{M\text{-}SIS},\kappa,8T_{\mathrm{op}}(b+1)\beta'} + \mathsf{adv}_{\mathsf{M\text{-}SIS},\kappa_1,2\beta'}, \tag{3}$$

where Q *is the number of queries made to the random oracle by the prover, and* r *is the number of witness vectors of rank* n *in the LaBRADOR statement output by the prover, respectively. The number of times that the knowledge extractor* $\mathcal{E}$ *invokes the prover is in expectation at most* $16r + 8 + Q(16r + 7)$.

Theorem 4.2 follows by application of the extractor from Theorem I.1, and a simple hybrid argument. For a discussion of the knowledge error of LaBRADOR when recursed, see Appendix I.5.

5 Adapting LaBRADOR for Aggregation

5.1 Changing the Modulus

We reformulate our verification constraints to be over a different modulus. From this point forward, we denote $\mathcal{R}_q$ with ring degree $d \in \{512, 1024\}$ and modulus $q = 12289$ as the *Falcon ring*, and $\mathcal{R}_{q'}$ with the same ring degree d but a larger modulus $q' > q$ as the *LaBRADOR ring.* To express the Falcon verification over the LaBRADOR ring, we first use the standard trick of lifting the equation to $\mathcal{R}$, by remembering the modular wrap-around. For each signature, we add an element $\boldsymbol{v}_i \in \mathcal{R}_{q'}$ to our witness, which should satisfy the equation

$$\boldsymbol{s}_{i,1} + \boldsymbol{h}_i \boldsymbol{s}_{i,2} + q\boldsymbol{v}_i - \boldsymbol{t}_i = \boldsymbol{0} \in \mathcal{R}, \tag{4}$$

without any modular reduction in the coefficients. If an equation holds over $\mathcal{R}$, then clearly the equation also holds modulo any modulus. However, with LaBRADOR, we can only prove that the equation holds modulo q', and then there are no guarantees that it also holds over $\mathcal{R}$. To prove that the equation holds over $\mathcal{R}$, we use an infinity norm check. Clearly, if the coefficients of our witness vectors are so small that they could not have caused a wrap around modulo q' in the equation, then the equation also holds over $\mathcal{R}$. The specifics of how we perform this norm check are described in the next subsection.

5.2 Norm Checks in the First Iteration

To prove that the aggregated signatures have ℓ_2-norm at most β, we follow the approach of [GHL22]. They show how to obtain an exact proof of smallness by combining the approximate proof of smallness from [LNS21] with a sum-of-squares proof. The approximate proof of smallness is specified so that it also ensures that there is no wraparound in the Falcon verification equation (4).

A) Four-Square Constraints. Proving that $\|\boldsymbol{s}_{i,1}\|_2^2 + \|\boldsymbol{s}_{i,2}\|_2^2 \le \beta^2$ is equivalent to proving that $\beta^2 - \|\boldsymbol{s}_{i,1}\|_2^2 - \|\boldsymbol{s}_{i,2}\|_2^2$ is non-negative. Lagrange's four-square theorem states that any non-negative integer can be written as the sum of four squares. Thus, we can find four integers $\varepsilon_{i,0}, \varepsilon_{i,1}, \varepsilon_{i,2}, \varepsilon_{i,3} \in \mathbb{Z}$ such that

$$\beta^2 - \|\boldsymbol{s}_{i,1}\|_2^2 - \|\boldsymbol{s}_{i,2}\|_2^2 = \varepsilon_{i,0}^2 + \varepsilon_{i,1}^2 + \varepsilon_{i,2}^2 + \varepsilon_{i,3}^2. \tag{5}$$

We add the four-square coefficients of the i-th signature to the witness as the coefficients of the polynomial $\boldsymbol{\varepsilon}_i = \varepsilon_{i,0} + \varepsilon_{i,1}X + \varepsilon_{i,2}X^2 + \varepsilon_{i,3}X^3 \in \mathcal{R}_{q'}$.

To formulate the four-square constraints in LaBRADOR, we make use of the relation between the conjugation automorphism σ_{-1} and the ℓ_2-norm in $\mathcal{R}_{q'}$. Recall that for any vector $\vec{\boldsymbol{a}} \in \mathcal{R}_{q'}^n$, $\|\vec{\boldsymbol{a}}\|_2^2 = \mathsf{ct}\left(\langle \sigma_{-1}(\vec{\boldsymbol{a}}), \vec{\boldsymbol{a}} \rangle\right) \bmod q'$. The idea is to provide $\boldsymbol{s}'_{i,1} = \sigma_{-1}(\boldsymbol{s}_{i,1})$, $\boldsymbol{s}'_{i,2} = \sigma_{-1}(\boldsymbol{s}_{i,2})$ and $\boldsymbol{\varepsilon}'_i = \sigma_{-1}(\boldsymbol{\varepsilon}_i)$ as part of the witness, so that we can compute the ℓ_2-norms directly in the dot product constraint. Then the four-square constraints (modulo q') can be expressed in LaBRADOR as

$$\mathsf{ct}\left(\boldsymbol{s}'_{i,1}\boldsymbol{s}_{i,1} + \boldsymbol{s}'_{i,2}\boldsymbol{s}_{i,2} + \boldsymbol{\varepsilon}'_i\boldsymbol{\varepsilon}_i - \beta^2\right) = 0 \bmod q'. \tag{6}$$

To prove that (6) implies (5), we need to prove two things. First, we need to prove that the new elements in the witness are of the correct form (B). Second, we need to prove that the coefficients of the witness elements involved in (6) are so small that they could not have caused a wrap-around modulo q' (C).

B) Constraints for the New Witness Elements. To prove that the new witness elements have been computed correctly, we add new dot product constraints, which check one coefficient at a time. The j-th coefficient of some $\boldsymbol{a} = \sum_{i=1}^{d-1} a_i X^i \in \mathcal{R}_{q'}$ can be directly verified through a dot product constraint as $\mathsf{ct}\left(\sigma_{-1}(X^j)\boldsymbol{a}\right) = a_j$. Each time we want to check that a_j is equal to some constant $b \in \mathbb{Z}_{q'}$, or we want to check that it is equal to the k-th coefficient of some other witness element $\boldsymbol{c} \in \mathcal{R}_{q'}$, we add a constraint of the form

$$\mathsf{ct}\left(\sigma_{-1}(X^j)\boldsymbol{a} - b\right) = 0 \bmod q' \quad \text{or} \quad \mathsf{ct}\left(\sigma_{-1}(X^j)\boldsymbol{a} - \sigma_{-1}(X^k)\boldsymbol{c}\right) = 0 \bmod q'. \tag{7}$$

For each $\boldsymbol{\varepsilon}_i$, we prove that it is a polynomial in $\mathcal{R}_{q'}$ of degree at most 4, by checking that the coefficients $\varepsilon_{i,4}, \ldots, \varepsilon_{i,d-1}$ are 0. For the $\boldsymbol{s}'_{i,1}, \boldsymbol{s}'_{i,2}, \boldsymbol{\varepsilon}'_i$, we observe that the conjugation automorphism induces a permutation of the coefficients of the input polynomial (up to a sign). For any $\boldsymbol{a} = a_0 + a_1 X + \cdots + a_{d-1}X^{d-1} \in \mathcal{R}_{q'}$, it holds $\sigma_{-1}(\boldsymbol{a}) = a_0 - a_{d-1}X - a_{d-2}X^2 - \cdots - a_1 X^{d-1}$. Therefore, we just check that each coefficient in $\boldsymbol{s}'_{i,j}$ matches the corresponding coefficient of $\boldsymbol{s}_{i,j}$, multiplied by the correct sign. The same goes for the $\boldsymbol{\varepsilon}'_i$. In total, we add roughly $4dN$ new constraints with these checks.

C) Approximate Proof of ℓ_∞-Smallness.

To prove that there is no wrap-around in the constraints (4) and (6) we use approximate proofs of ℓ_∞-smallness. For this, we use the built in projection step in LaBRADOR for approximate ℓ_2-smallness. This works because the ℓ_2-norm of the witness is an upper bound on its ℓ_∞-norm. In [GHL22] a protocol for approximate proofs of ℓ_∞-smallness is presented, which can be adapted to our setting, but it concretely ended up giving us slightly larger proofs.

The task at hand is therefore to configure the projection protocol to be our approximate proof of ℓ_∞-smallness. First, we need to find a ℓ_∞-norm bound β_∞ for the witness that would guarantee that there could be no wrap-around. Next, we need to find a ℓ_2-norm upper bound β_2 for the witness of the honest prover. For both security levels $\lambda \in \{128, 256\}$, the projection is a proof that the ℓ_2-norm of the witness of the prover is at most $\sqrt{128/30}\beta_2$. For completeness we want $\sqrt{128/30}\beta_2 \leq \beta_\infty$, implying that the witness indeed has ℓ_∞-norm at most β_∞. Lastly, we need to derive a lower bound for how small the LaBRADOR modulus q' can be while still ensuring that the projection proof is complete and sound. To reduce the size of q', we actually do two projections, one for the $\boldsymbol{v}_i$ and one for the rest. Hence, we need to find a β_∞ and β_2 for each projection, and set q' such that both are sound proofs of approximate ℓ_∞-smallness.

We begin by deriving suitable ℓ_∞-norm bounds for the witness elements. For the constraints of interest, we want to argue that before reducing modulo q', the infinity norm of the constraint evaluated on the witness is strictly less than $q'/2$. If the constraint is satisfied modulo q', it must also be satisfied over the integers.

Let us first consider the i-th four-square constraint (6). It states that the sum of the square of each coefficient of $\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}, \boldsymbol{\varepsilon}_i$ should be equal to β^2 modulo q'. If we require that each of the $2d+4$ coefficients has ℓ_∞-norm strictly less than $\sqrt{q'/(2(2d+4))}$, this sum of squares is strictly less than $q'/2$. Then the sum is also equal to β^2 over the integers. Hence, we need

$$\left\| \|_{i=1}^{N} (\boldsymbol{s}_{i,1} \| \boldsymbol{s}_{i,2} \| \boldsymbol{s}'_{i,1} \| \boldsymbol{s}'_{i,2} \| \boldsymbol{\varepsilon}_i \| \boldsymbol{\varepsilon}'_i) \right\|_\infty < \sqrt{\frac{q'}{2(2d+4)}}.$$

Next we consider the i-th Falcon verification constraint (4). It states that the sum of the three terms $\boldsymbol{s}_{i,1}$, $\boldsymbol{h}_i \boldsymbol{s}_{i,2}$ and $q\boldsymbol{v}_i$ should be equal to $\boldsymbol{t}_i$ modulo q'. By requiring that the ℓ_∞-norm of each term is strictly less than $q'/6$, their sum has infinity norm strictly less than $q'/2$. By Lemma 2.1 we get that $\|\boldsymbol{h}_i \boldsymbol{s}_{i,2}\|_\infty \leq d \|\boldsymbol{h}_i\|_\infty \|\boldsymbol{s}_{i,2}\|_\infty \leq dq \|\boldsymbol{s}_{i,2}\|_\infty$. Hence, we need

$$\|\boldsymbol{s}_{i,1}\|_\infty < q'/6, \qquad \|\boldsymbol{s}_{i,2}\|_\infty < q'/6dq, \qquad \|\boldsymbol{v}_i\|_\infty < q'/6q.$$

Notice that we have two ℓ_∞-norm bound requirements for $\boldsymbol{s}_{i,1}$ and $\boldsymbol{s}_{i,2}$. However, we see later that q' must be so large that the bound from the four-square constraint is the most restrictive. Thus, we arrive at the final β_∞ bound for each of our projections:

$$\left\| \|_{i=1}^{N} (\boldsymbol{s}_{i,1} \| \boldsymbol{s}_{i,2} \| \boldsymbol{s}'_{i,1} \| \boldsymbol{s}'_{i,2} \| \boldsymbol{\varepsilon}_i \| \boldsymbol{\varepsilon}'_i) \right\|_\infty < \beta_\infty^{(1)} = \sqrt{\frac{q'}{2(2d+4)}},$$
$$\|\boldsymbol{v}_1 \mid \ldots \| \boldsymbol{v}_N\|_\infty < \beta_\infty^{(2)} = q'/6q.$$

The second step in specifying our approximate proof of ℓ_∞-smallness is to find the β_2 for each projection. For the first projection, we have $\beta_2^{(1)} = 2\beta\sqrt{N}$. Here, the factor 2 comes from the fact that for all $\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}$

and $\boldsymbol{\varepsilon}_i$ we also include the respective conjugate in the witness. For the projection of the $\boldsymbol{v}_i$, we have by Lemma 2.1 that

$$\|\boldsymbol{v}_i\|_2 = \frac{1}{q}\|\boldsymbol{t}_i - \boldsymbol{s}_{i,1} - \boldsymbol{h}_i\boldsymbol{s}_{i,2}\|_2 \leq \frac{1}{q}\left(\sqrt{d}q + \beta + dq\beta\right).$$

In Falcon $\beta < q$, so that $\|\boldsymbol{v}_i\|_2 < 1 + \sqrt{d} + d\beta$. Hence, the second projection has $\beta_2^{(2)} = (1 + \sqrt{d} + d\beta)\sqrt{N}$.

Finally, we analyze the requirements on q' for the approximate ℓ_∞-smallness protocol to be complete and sound. For each projection $k \in \{1, 2\}$ and security level λ, there are two requirements on q':

1. $\sqrt{\lambda}\beta_2^{(k)} \leq q'/C_1$. This is the condition for using the Johnson-Linden -strauss Lemma 2.2. When this lemma is applicable, the projection step is a proof that the ℓ_2-norm is at most $\sqrt{\lambda/C_2}\beta_2^{(k)}$.
2. $\sqrt{\lambda/C_2}\beta_2^{(k)} \leq \beta_\infty^{(k)}$. This means that the projection step is a proof that the witness has infinity norm at most β_∞.

Concretely comparing the constraints on q' when using Falcon-512 or Falcon-1024, we get that the second condition for the first projection is the most restrictive. With our Johnson-Lindenstrauss parameters, we need

$$q' > (1024/15)(d + 2)\beta^2 N. \tag{8}$$

In practice, this bound keeps q' at a reasonable size for implementations. For fast 64-bit lattice implementations, we want q' to be at most a couple of bits less than 64-bits. For Falcon-512, the constraint (8) translates to $q' > 2^{40.12}N$ and for Falcon-1024 to $q' > 2^{42.16}N$. In practice, it seems reasonable to assume that $N \leq 2^{20}$. At $N = 2^{20}$ we need a 61-bit modulus for Falcon-512 and a 63-bit modulus for Falcon-1024, which is at the top end of the allowable range. At the more realistic $N = 2^{10}$, q' only needs to be 51-bits (respectively, 53-bits) long.

5.3 Reformulating the Constraints for a Better Recursion

Until now we have payed little attention to the relationship between the number of witnesses and their rank. This relationship has an impact on prover runtime and how fast LaBRADOR iterations converge towards the base case. In Appendix F.1, we show how to reshape our witness such that we get $r = O(\sqrt{N})$ witness vectors of rank $n = N$, giving us better runtimes and slightly smaller proofs. The tricky part of reshaping the witness is that we must still be able to formulate the four-square constraints. All other constraints are linear in the witness elements, so can be easily reformulated to fit the new witness. However, the four-square constraints are *quadratic* constraints. To formulate them, one needs to compute $\|(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2})\|_2^2$ in LaBRADOR as the inner product of witness vectors. We solve this problem with a new padding scheme. See Appendix F.2 for the final constraints.

5.4 Working over Subring

Finally, we reformulate our verification constraints to be over a subring. In particular, we move from the ring $\mathcal{R}$ of degree d to a subring $\mathcal{S}$ of smaller degree $d' = d/c$ which improves efficiency. To this end, we make use of the bijection $\phi\colon \mathcal{R}_{q'}^n \to \mathcal{S}_{q'}^{c \cdot n}$ defined in Section 2.2. It suffices to show that the bijection is compatible with the LaBRADOR constraint system. Let $\mathcal{F}$ denote the full dot product constraints and $\mathcal{F}'$ the constant term constraints resulting from the discussion above on aggregating Falcon signatures, as summarized in Appendix F.2. As it is a bijection, we know that only the zero element is mapped to zero. More importantly, the map is norm-preserving. Then, any tuple $(\mathcal{F}, \mathcal{F}', \beta)$ over $\mathcal{R}_q$ defines a tuple $(\phi(\mathcal{F}), \phi(\mathcal{F}'), \beta)$ over $\mathcal{S}_q^c$ such that

$$\begin{aligned}
\phi(f)(\phi(\vec{\boldsymbol{w}}_1), \ldots, \phi(\vec{\boldsymbol{w}}_r)) = \mathbf{0} \in \mathcal{S}_q^c \qquad & \forall \phi(f) \in \phi(\mathcal{F}), \\
\mathsf{ct}\left(\phi(f')(\phi(\vec{\boldsymbol{w}}_1), \ldots, \phi(\vec{\boldsymbol{w}}_r))\right) = \mathbf{0} \in \mathbb{Z}_q^c \qquad & \forall \phi(f') \in \phi(\mathcal{F}'), \\
\sum_{i=1}^{r} \|\phi(\vec{\boldsymbol{w}}_i)\|_2^2 \leq \beta^2. &
\end{aligned}$$

As we see in Section 6.1 below, this last technique helps reducing the proof sizes by at least a multiplicative factor 2.

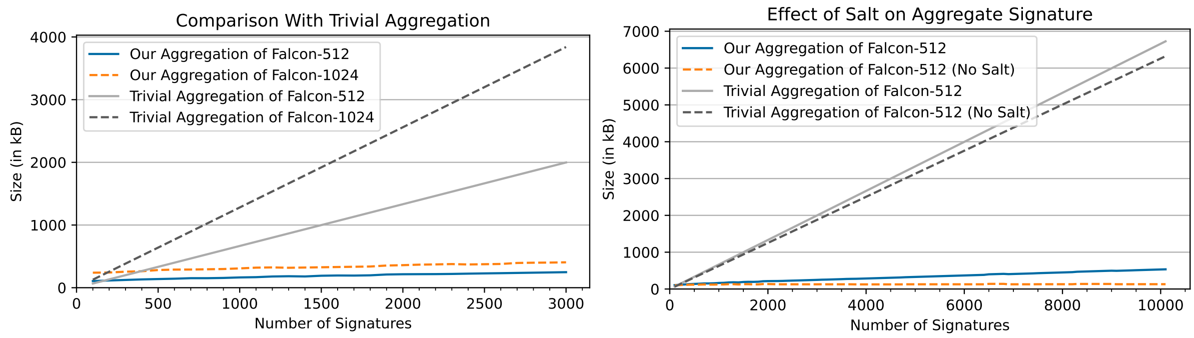


Fig. 3: The size in kB of our aggregate signature compared to the trivial aggregation through concatenation, for both Falcon-512 and Falcon-1024 (Left), and compared to the trivial aggregation through concatenation for Falcon-512, both with and without salt (Right). With salt corresponds to the original Falcon scheme, and without salt corresponds to a deterministic version of Falcon, not proposed for standardization. On the left graph we only show sizes for up to 3000 signatures to allow easier comparisons for small numbers of signatures.

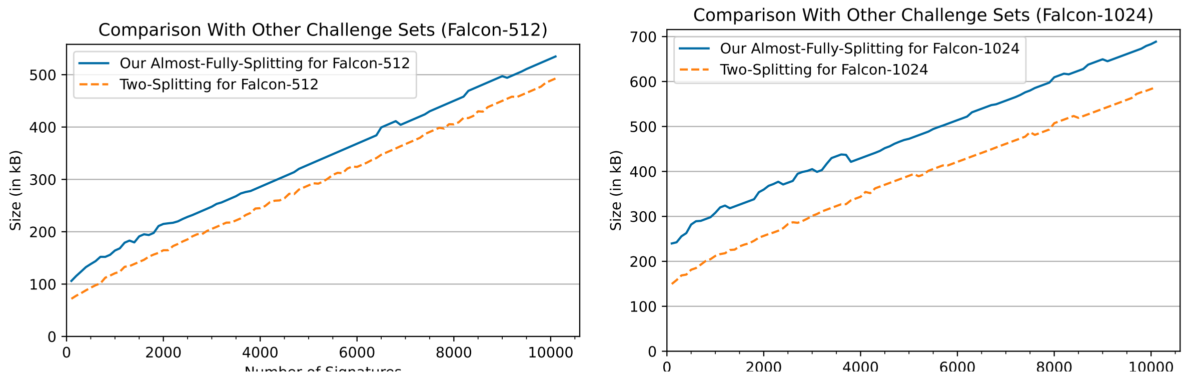


Fig. 4: The size in kB of our aggregate signature scheme compared to the sizes for different choices of challenge sets, for both Falcon-512 and Falcon-1024.

6 Estimates

fullversion The program code to compute the numbers and plots in this section can be found and assessed in our online repository at `https://github.com/dfaranha/aggregate-falcon`.

Overall, our aggregate signature scheme for Falcon-512 signatures guarantees a security level of 121 bits, whereas the aggregate signature scheme for Falcon-1024 signatures guarantees 249 bits of security. We give a detailed explanation how this is derived in Appendix E.

6.1 Estimates of Proof Sizes

We tailored all choices of parameters to allow for maximal $N = 10\,000$ signatures to be aggregated. We highlight that if setting up LaBRADOR for aggregating significantly smaller or larger number of Falcon signatures, we recommend re-running our scripts to derive different parameter sets. Setting everything up for a much smaller N would also lead to significantly smaller AS.

Comparison With Trivial Concatenation. Figure 3 (Left) provides estimates for the proof sizes of our aggregate signature (with salts) for both Falcon parameter regimes and compares it with the trivial solution of concatenating all individual signatures. The parameter regimes Falcon-512 and Falcon-1024 correspond to the parameter sets proposed in the specifications of Falcon, as recalled in Table 2, as well as our choice of challenge sets as described in Section D.4. We observe that for both parameter regimes starting from ca. 200 signatures, our aggregate signature is shorter than the trivial solution. Whereas both the trivial aggregation and our aggregate signature (due to the salts) grow linearly with the number of signatures, one can clearly see ours provides significantly shorter aggregate signatures.

Effect of the Salt. As the salt of every individual Falcon signature is included in the final aggregate signature (cf. the construction in Section C.2), the size of the aggregate signature is linear in the number N. Thus, our scheme is not succinct from an asymptotic point of view. However, the salt in Falcon consists of 320 bits, which is only a small part of the complete signature. The rather small effect of the salt is depicted in Figure 3 (Right). Only for rather large numbers of signatures N, the effect is significant. For example, for $\lambda = 128$, starting from $N = 3200$, the size of σ_{agg} with salts is twice as big as the size of σ_{agg} without salts. For $\lambda = 256$, this is only the case starting from $N = 6700$.

We highlight that our goal was to use the native language of LaBRADOR to aggregate Falcon signatures. Hence, instantiating the random oracle with a concrete hash function, adding the salts to the witness and moving the hash evaluation *inside* the SNARK is not an option for us, as the hash evaluation cannot be expressed directly in the native LaBRADOR language. One would need to go to the R1CS constraint system. Thus, the only way to avoid sending the salt is to move to a *deterministic* version of Falcon. As we think that this is not a very practical solution (for example, how to manage software updates? And how to guarantee the same floating point arithmetic on all devices?), we focus on the original version of Falcon with salt. For illustration purposes, we sometimes also provide the sizes of our aggregate signature if no salts would be included, i.e. for deterministic Falcon.

Challenge Sets. As explained in Section 3, there are different choices for the splitting behavior of the ring $\mathcal{S}$ modulo q'. Overall, we distinguish two settings: two-splitting rings (as in the original LaBRADOR [BS23]) and almost-fully-splitting rings (as in our protocol).

In the almost-fully-splitting case, the probability that challenges and challenge differences are invertible depends on the well-spreadness of the challenge space (cf. Lemma 3.1). To derive parameters for the challenges (weight and infinity norm bound), we adapted the (heuristic) SageMath code from [ESZ22]. In order to make the probability of non-invertibility as small as $2^{-\lambda}$, we require a larger weight than in the two-splitting case. This explains why the curve of the two-splitting case is slightly below the curve of the almost-fully-splitting case. In the two-splitting case, due to [LS18, Cor. 1.2], any non-zero ternary challenge (difference) is invertible if $q' > 8$, so parameters must only account for making the challenge space $\mathcal{C}$ big enough.

Overall, our numbers in Figure 4 show that going from two-splitting to almost-fully-splitting comes at some small costs, while now allowing for much faster computations. In the next section, we argue that the benefit with respect to computation efficiency is much more significant than the loss in aggregate signature sizes. If the curious reader is wondering why the lines in Figure 4 are not monotonically increasing, we would like to mention that our global recursion strategy (as explained in Section 5.3) might not be *optimal* for a specific number of signatures N. We think it is possible to obtain a smooth curve if the recursion strategy is locally tailored for a specific N. We leave this up to future work.

6.2 Estimates of Running Times for Polynomial Arithmetic

In section 3 we proposed to modify the challenge space of LaBRADOR, to rely on an almost-fully-splitting ring $\mathcal{R}_{q'} = \prod_{i=1}^{l} \mathbb{Z}_{q'}[X]/\langle X^\delta - \zeta_i \rangle$ with small δ. Since LaBRADOR extensively uses polynomial multiplications both in proof generation and verification, the choice of ring and challenge space determines the overall performance. A major factor is supporting efficient polynomial multiplication, for example through the Number Theoretic Transform (NTT) [LS19].

In general, when the splitting ratio δ for a fixed ring degree d' is halved, one additional level of the NTT transform is computed, and the degree of the resulting polynomials when performing multiplication in the NTT domain is also halved. This means that the cost of polynomial multiplication decreases roughly by 2, assuming that a simple Schoolbook algorithm is used for multiplying the d'/δ polynomials of degree δ remaining in the last level at cost $O(\delta^2)$. For a fully-splitting ring, polynomial multiplication becomes point-wise multiplication. In other words, dividing the splitting factor by 4 is roughly equivalent to doubling the size of the modulus q', also assuming Schoolbook multiplication in $\mathbb{Z}_{q'}$.

We validated the estimated performance behavior with a proof-of-concept implementation of the NTT evaluations and polynomial multiplication in the NTT domain for various choices of $\mathcal{R}_{q'}$. Even though we expect polynomial multiplication to dominate proof generation and verification, we also report the cost of evaluating the NTT transform. The implementation for the two-splitting case was performed using the FLINT v2.9 library, and code adapted from [FWK23] for the other NTT-friendly parameters. Performance figures were collected using GCC 13.2.1 in a 64-bit Intel Core i7-7700 Kaby Lake CPU running at 3.60GHz, with TurboBoost turned off to reduce randomness in the runtime. We first experimentally observed that a field multiplication modulo a double-word q' using Montgomery modular multiplication is approximately 7 times the latency of a multiplication modulo a single-word q' using the signed Montgomery variant. With this ratio at hand, we were able to scale the latency from single-word to double-word, since multiplication in $\mathbb{Z}_{q'}$ is the performance-critical operation for both the NTT transforms and polynomial multiplication in the NTT domain.

Table 1 presents the clock cycles for polynomial arithmetic with different δ and subring degree d'. For Falcon-512, we considered the cases $\delta = 4$ and fully-splitting with 128-bit q', such that $(q')^\delta$ would be at least 2^λ. The estimates above suggest similar latency for both parameters, but we observe a 40% difference favoring the fully-splitting case with a double-word modulus, which does not compensate for the larger proofs. For Falcon-1024, we considered the cases $\delta = 4$ and $\delta = 8$, which favor the larger splitting ratio by more than a 2-factor, in accordance to the general estimate above. From these empirical results, we conclude that $\delta = 4$ is most suitable for Falcon-512, whereas $\delta = 8$ is best for Falcon-1024 when both execution time and proof size are taken into account. The code can be found in the supplementary material and in our online anonymous repository.

Open problem. The bijection between rings, described in Section 5.4, could be applied between any pair of recursive LaBRADOR iterations. The final proof sizes largely depend on the last iteration and the degree of the ring it uses. Therefore, the ideal space-time tradeoff might be found by first using a ring with greater splitting factor for faster multiplications, and then moving to two-splitting for the last iteration(s) getting the smallest ring degree. We leave a full analysis as future work, noting it would likely require an implementation of LaBRADOR.

Table 1: Timings in clock cycles for polynomial arithmetic (multiplication and NTT when applicable) for efficient choices of subring with degree d'. Numbers in italic correspond to the best trade-off between execution time and proof size.

	Falcon-512		Falcon-1024	
Splitting regime / Operation	Mult.	(inv) NTT	Mult.	(inv) NTT
Two ($d' = d/8$)	24,700	–	67,435	–
Almost-fully ($\delta = 8, d' = d/4$)	–	–	*29,412*	*13,148*
Almost-fully ($\delta = 4, d' = d/4$)	*6,284*	*8,089*	91,343	114,408
Fully ($d' = 64$)	3,815	37,443	–	–

Acknowledgment

The authors are grateful to Gregor Seiler for answering our technical questions about LaBRADOR. This research project was mainly done while Katharina Boudgoust was affiliated with Aarhus University. The work was supported by the Danish Independent Research Council under projects 0165-00107B (C3PO), 1026-00350B (RENAIS) and 2064-00016B (YOSO), as well as by the Protocol Labs Research Grant Program RFP-013 and PL-RGP1-2021-064, and the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC). The authors are grateful to Ngoc Khanh Nguyen for helpful feedback on earlier versions of this work.

This paper was prepared in part for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

ACK21. T. Attema, R. Cramer, and L. Kohl. A compressed Σ-protocol theory for lattices. In *CRYPTO 2021, Part II*, vol. 12826 of *LNCS*, pp. 549–579, Virtual Event, 2021. Springer, Heidelberg.

ACL+22. M. R. Albrecht, V. Cini, R. W. F. Lai, G. Malavolta, and S. A. K. Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In *CRYPTO 2022, Part II*, vol. 13508 of *LNCS*, pp. 102–132. Springer, Heidelberg, 2022.

AFK22. T. Attema, S. Fehr, and M. Klooß. Fiat-shamir transformation of multi-round interactive proofs. In *TCC 2022, Part I*, vol. 13747 of *LNCS*, pp. 113–142. Springer, Heidelberg, 2022.

AL21. M. R. Albrecht and R. W. F. Lai. Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In *CRYPTO 2021, Part II*, vol. 12826 of *LNCS*, pp. 519–548, Virtual Event, 2021. Springer, Heidelberg.

ALS20. T. Attema, V. Lyubashevsky, and G. Seiler. Practical product proofs for lattice commitments. In *CRYPTO 2020, Part II*, vol. 12171 of *LNCS*, pp. 470–499. Springer, Heidelberg, 2020.

AMAB+23. C. Aguilar-Melchor, M. R. Albrecht, T. Bailleux, N. Bindel, J. Howe, A. Hülsing, D. Joseph, and M. Manzano. Batch signatures, revisited. Cryptology ePrint Archive, Report 2023/492, 2023. https://eprint.iacr.org/2023/492.

BBB+18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pp. 315–334. IEEE Computer Society Press, 2018.

BCC+16. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 327–357. Springer, Heidelberg, 2016.

BCMS20. B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. Recursive proof composition from accumulation schemes. In *TCC 2020, Part II*, vol. 12551 of *LNCS*, pp. 1–18. Springer, Heidelberg, 2020.

BCPS18. A. Bishnoi, P. L. Clark, A. Potukuchi, and J. R. Schmitt. On zeros of a polynomial in a finite grid. *Comb. Probab. Comput.*, 27(3):310–333, 2018.

BDF+11. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In *ASIACRYPT 2011*, vol. 7073 of *LNCS*, pp. 41–69. Springer, Heidelberg, 2011.

BDN18. D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *ASIACRYPT 2018, Part II*, vol. 11273 of *LNCS*, pp. 435–464. Springer, Heidelberg, 2018.

BF23. B. Bünz and B. Fisch. Multilinear schwartz-zippel mod N with applications to succinct arguments. In *TCC 2023*, Lecture Notes in Computer Science. Springer-Verlag, 2023.

BGLS03. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT 2003*, vol. 2656 of *LNCS*, pp. 416–432. Springer, Heidelberg, 2003.

BK20. D. Boneh and S. Kim. One-time and interactive aggregate signatures from lattices. *preprint*, 2020.

BN06. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 2006*, pp. 390–399. ACM Press, 2006.

BNN07. M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In *ICALP 2007*, vol. 4596 of *LNCS*, pp. 411–422. Springer, Heidelberg, 2007.

BR21. K. Boudgoust and A. Roux-Langlois. Compressed linear aggregate signatures based on module lattices. Cryptology ePrint Archive, Report 2021/263, 2021. https://eprint.iacr.org/2021/263.

BRL23. K. Boudgoust and A. Roux-Langlois. Overfull: Too Large Aggregate Signatures Based on Lattices. *The Computer Journal*, p. bxad013, 2023.

BS23. W. Beullens and G. Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In *CRYPTO 2023, Part V*, vol. 14085 of *LNCS*, pp. 518–548. Springer, Heidelberg, 2023.

BT23. K. Boudgoust and A. Takahashi. Sequential half-aggregation of lattice-based signatures. In *ESORICS 2023*, Lecture Notes in Computer Science. Springer, 2023.

BTT22. C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In *CRYPTO 2022, Part II*, vol. 13508 of *LNCS*, pp. 276–305. Springer, Heidelberg, 2022.

Che23. Y. Chen. sfDualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In *CRYPTO 2023, Part V*, vol. 14085 of *LNCS*, pp. 716–747. Springer, Heidelberg, 2023.

CJJ22. A. R. Choudhuri, A. Jain, and Z. Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd FOCS*, pp. 68–79. IEEE Computer Society Press, 2022.

Cor02. J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In *EUROCRYPT 2002*, vol. 2332 of *LNCS*, pp. 272–287. Springer, Heidelberg, 2002.

DGKV22. L. Devadas, R. Goyal, Y. Kalai, and V. Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd FOCS*, pp. 1057–1068. IEEE Computer Society Press, 2022.

DHSS20. Y. Doröz, J. Hoffstein, J. H. Silverman, and B. Sunar. MMSAT: A scheme for multimessage multiuser signature aggregation. Cryptology ePrint Archive, Report 2020/520, 2020. https://eprint.iacr.org/2020/520.

DOTT22. I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, 2022.

EB14. R. El Bansarkhani and J. Buchmann. Towards lattice based aggregate signatures. In *AFRICACRYPT 14*, vol. 8469 of *LNCS*, pp. 336–355. Springer, Heidelberg, 2014.

ENS20. M. F. Esgin, N. K. Nguyen, and G. Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT 2020, Part II*, vol. 12492 of *LNCS*, pp. 259–288. Springer, Heidelberg, 2020.

ESLR23. M. F. Esgin, R. Steinfeld, D. Liu, and S. Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. In *CRYPTO 2023, Part V*, vol. 14085 of *LNCS*, pp. 484–517. Springer, Heidelberg, 2023.

ESZ22. M. F. Esgin, R. Steinfeld, and R. K. Zhao. MatRiCT$^+$: More efficient post-quantum private blockchain payments. In *2022 IEEE Symposium on Security and Privacy*, pp. 1281–1298. IEEE Computer Society Press, 2022.

FHSZ23. N. Fleischhacker, G. Herold, M. Simkin, and Z. Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 386–400, 2023.

FMN23. G. Fenzi, H. Moghaddas, and N. K. Nguyen. Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. Cryptology ePrint Archive, Paper 2023/846, 2023. https://eprint.iacr.org/2023/846.

FN16. D. Fiore and A. Nitulescu. On the (in)security of SNARKs in the presence of oracles. In *TCC 2016-B, Part I*, vol. 9985 of *LNCS*, pp. 108–138. Springer, Heidelberg, 2016.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, vol. 263 of *LNCS*, pp. 186–194. Springer, Heidelberg, 1987.

FSZ22. N. Fleischhacker, M. Simkin, and Z. Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In *ACM CCS 2022*, pp. 1109–1123. ACM Press, 2022.

FWK23. V. Farzaliyev, J. Willemson, and J. K. Kaasik. Improved lattice-based mix-nets for electronic voting. *IET Inf. Secur.*, 17(1):18–34, 2023.

GHL22. C. Gentry, S. Halevi, and V. Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *EUROCRYPT 2022, Part I*, vol. 13275 of *LNCS*, pp. 458–487. Springer, Heidelberg, 2022.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pp. 197–206. ACM Press, 2008.

GR06. C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In *PKC 2006*, vol. 3958 of *LNCS*, pp. 257–273. Springer, Heidelberg, 2006.

HKW15. S. Hohenberger, V. Koppula, and B. Waters. Universal signature aggregators. In *EUROCRYPT 2015, Part II*, vol. 9057 of *LNCS*, pp. 3–34. Springer, Heidelberg, 2015.

HPS98. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, vol. 1423 of *LNCS*, pp. 267–288. Springer, Heidelberg, 1998.

JRLS23. C. Jeudy, A. Roux-Langlois, and O. Sanders. Revisiting preimage sampling for lattices. Cryptology ePrint Archive, Paper 2023/446, 2023. Version 23.05.2023 https://eprint.iacr.org/archive/2023/446/20230516:150953.

JRS23. C. Jeudy, A. Roux-Langlois, and O. Sanders. Phoenix: Hash-and-sign with aborts from lattice gadgets. Cryptology ePrint Archive, Report 2023/446, 2023. https://eprint.iacr.org/2023/446.

LMRS04. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *EUROCRYPT 2004*, vol. 3027 of *LNCS*, pp. 74–90. Springer, Heidelberg, 2004.

LNP22. V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In *CRYPTO 2022, Part II*, vol. 13508 of *LNCS*, pp. 71–101. Springer, Heidelberg, 2022.

LNPS21. V. Lyubashevsky, N. K. Nguyen, M. Plançon, and G. Seiler. Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In *ASIACRYPT 2021, Part IV*, vol. 13093 of *LNCS*, pp. 218–248. Springer, Heidelberg, 2021.

LNS21. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *PKC 2021, Part I*, vol. 12710 of *LNCS*, pp. 215–241. Springer, Heidelberg, 2021.

LS15. A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75:565–599, 2015.

LS18. V. Lyubashevsky and G. Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT 2018, Part I*, vol. 10820 of *LNCS*, pp. 204–224. Springer, Heidelberg, 2018.

LS19. V. Lyubashevsky and G. Seiler. NTTRU: Truly fast NTRU using NTT. *IACR TCHES*, 2019(3):180–201, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8293.

LW15. V. Lyubashevsky and D. Wichs. Simple lattice trapdoor sampling from a broad class of distributions. In *PKC 2015*, vol. 9020 of *LNCS*, pp. 716–730. Springer, Heidelberg, 2015.

Lyu09. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, vol. 5912 of *LNCS*, pp. 598–616. Springer, Heidelberg, 2009.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 738–755. Springer, Heidelberg, 2012.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 700–718. Springer, Heidelberg, 2012.

MR09. D. Micciancio and O. Regev. Lattice-based cryptography. *Post-quantum cryptography*, pp. 147–191, 2009.

Ngu22. N. K. Nguyen. *Lattice-Based Zero-Knowledge Proofs Under a Few Dozen Kilobytes*. PhD thesis, ETH Zurich, 2022. https://www.research-collection.ethz.ch/handle/20.500.11850/574844.

NS22. N. K. Nguyen and G. Seiler. Practical sublinear proofs for R1CS from lattices. In *CRYPTO 2022, Part II*, vol. 13508 of *LNCS*, pp. 133–162. Springer, Heidelberg, 2022.

PFH+22. T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

TS23. T. Tomita and J. Shikata. Compact signature aggregation from module-lattices. Cryptology ePrint Archive, Report 2023/471, 2023. https://eprint.iacr.org/2023/471.

Val08. P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC 2008*, vol. 4948 of *LNCS*, pp. 1–18. Springer, Heidelberg, 2008.

Wik21. D. Wikström. Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265, 2021. https://eprint.iacr.org/2021/1265.

Win14. A. Winkelbauer. Moments and absolute moments of the normal distribution, 2014.

WW19. Z. Wang and Q. Wu. A practical lattice-based sequential aggregate signature. In *ProvSec 2019*, vol. 11821 of *LNCS*, pp. 94–109. Springer, Heidelberg, 2019.

WW22. B. Waters and D. J. Wu. Batch arguments for sfNP and more from standard bilinear group assumptions. In *CRYPTO 2022, Part II*, vol. 13508 of *LNCS*, pp. 433–463. Springer, Heidelberg, 2022.

WW23. H. Wee and D. J. Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. In *ASIACRYPT 2023*, Lecture Notes in Computer Science. Springer-Verlag, 2023.

A Related Work

A.1 Aggregate Signatures from Lattices

Aggregate Signatures and BARG/SNARK-based Solutions. Boneh et al. [BGLS03] introduced the notion of aggregate signatures for the first time, followed by a number of efficient constructions based on pre-quantum assumptions e.g. [GR06, BNN07]. Several generic constructions of AS based on non-interactive arguments exist in the literature, although none of them evaluate concrete efficiency if instantiated with the NIST finalists. Hohenberger, Koppula, and Waters [HKW15] construct AS from iO and the RSA assumption. Waters and Wu [WW22] propose BARGs for NP from standard pairing-based assumptions and use BARGs to construct compact AS in the standard model. To prove the security of AS, they show *somewhere* knowledge soundness of BARG is sufficient, meaning that an extractor only needs to obtain a single witness for one of the statements $\mathsf{x}_1, \ldots, \mathsf{x}_N$ chosen by the adversary. Their generic AS construction could be instantiated with LWE-based BARG of [CJJ22]. Devadas et al. [DGKV22] generalize BARG to *multi-hop* BARG, which allows aggregating multiple batch proofs for NP statements. Their construction of multi-hop BARGs from LWE can be applied to instantiate multi-hop AS in the standard model, where an aggregator can combine possibly aggregated signatures. Outside the standard model, Tomita and Shikata [TS23] observe that LaBRADOR can be used in a straightforward manner to instantiate lattice-based AS in the random oracle model, by translating the verification condition of the base signature scheme to R1CS.

Our work takes a different angle from these generic feasibility results in that (1) we strive to optimize the LaBRADOR relation and challenge space in order to natively support the verification equations of the standardized Falcon scheme, instead of naively converting them into a boolean circuit or R1CS, (2) we provide concrete size estimates of our AS construction, (3) we present a general framework for analyzing Fiat-Shamir AoK from multi-round protocols including LaBRADOR, and (4) we explicitly prove a signing oracle for hash-then-sign signatures does not interfere with witness extraction to formally conclude security of SNARK-based AS (in the random oracle model).

Aggregate Signatures tailored to GPV. To the best of our knowledge, only a few GPV-based AS exist in the literature. Jeudy, Roux-Langlois, and Sanders [JRLS23] present AS constructed from the Micciancio-Peikert trapdoor [MP12] and Lyubashevsky-Wichs Gaussian sampler [LW15]. Since their aggregation strategy highly exploits particulars of [MP12]-based GPV, it is currently unclear how a similar approach extends to an NTRU-based instantiation of GPV including Falcon. By allowing signers interact with each other in a round-robin fashion, one can obtain a *sequential* aggregate signature (SAS) [LMRS04]. As mentioned earlier, two existing SAS based on GPV [WW19, EB14] only offer very limited compression rates.

Aggregate Signatures tailored to Fiat-Shamir with Aborts. Within the Fiat-Shamir with Aborts paradigm [Lyu09, Lyu12], MMSAT [DHSS20] was proposed as a *half*-aggregate signature, compressing only half of the two signature components. Boudgoust and Roux-Langlois [BRL23, BR21] point out a flaw in MMSAT and present a secure variant of MMSAT assuming Module-SIS, Module-LWE and ROM, but they also observe that the size of aggregate signature is larger than the trivial concatenation. Very recently, Boudgoust an Takahashi [BT23] presented a FS-based SAS that outputs an aggregate signature smaller than the naive concatenation. Similar to GPV-based SAS, their concrete compression rate is still quite limited ($\approx 99\%$) if instantiated with Dilithium parameter sets.

Allowing interaction between signers, one can construct Fiat-Shamir-based AS by applying a generic conversion method to an interactive multi-signature (where all the signers sign the same message) [BN06, BK20]. Thus, a line of work on lattice-based multi-signatures e.g. [DOTT22, BTT22, Che23] could also be turned into interactive AS. This generic method does not fit in the typical use cases of AS such as a certificate chain, because it has to ask the signers to synchronize with each other in advance to agree on all N messages to be signed.

Synchronized Aggregate Signatures. Squirrel [FSZ22] and its recent optimization Chipmunk [FHSZ23] can be viewed as lattice-based AS but follow an entirely different paradigm than GPV or FS. Although they provide concretely efficient constructions, Squirrel and Chipmunk come with limitations: (1) the constructions are only proven secure in a "synchronized" model where signers are only allowed to produce a single signature in each time step, and (2) they only allow a signer to produce a bounded number of signatures for a fixed public key. Our approach to AS does not suffer from these drawbacks.

Related Notions. Batch Signature (BS) studied by Aguilar-Melchor et al. [AMAB+23] is a different primitive than AS. While AS asks an aggregator to compress N individually generated signatures, BS requests a single signer to generate a compact signature on N messages with the same signing key. AS and BS share somewhat similar goals in that BS also reduces bandwidth and verification complexity, but it requires modification in the signing operations. In contrast, AS typically introduces a customized verification algorithm while the sign algorithm is unchanged.

A.2 Concrete Analysis of Fiat-Shamir

Attema, Cramer, and Kohl [ACK21] generalized special soundness for Σ-protocol to $\mathbf{K} = (k_1, \ldots, k_\mu)$-special soundness for $(2\mu+1)$-round public-coin protocols and presented an improved analysis of knowledge extractor in the interactive setting. Attema, Fehr, and Klooß [AFK22] and Wikström [Wik21] concurrently proved concrete knowledge error of Fiat-Shamir AoK constructed from $\mathbf{K}$-special sound protocols. Fenzi and Nguyen [FMN23] generalized the result of [AFK22] to account for coordinate-wise special sound (CWSS) protocols. Nguyen [Ngu22, 5.1.3,8.5.1.3] analyzes knowledge soundness of their multi-round interactive protocols using a similar extraction strategy to ours (i.e., extract a candidate witness w, and test the validity of w with freshly sampled challenges). Although our analysis of "extract and test" extraction in the Fiat-Shamir setting may partially resemble [LNP22] and [BF23], we highlight the advantages of our approach in detail.

The Extractor of [LNP22]. Lyubashevsky, Nguyen and Plançon [LNP22, Appendix B] take a similar approach to ours to analyze Fiat-Shamir AoK derived from a 9-round protocol. Although they extend the abstract sampling game of [AFK22] to accommodate probabilistic tests post-extraction, their analysis focuses on the special case where only two challenge values are needed for every round, assuming *always* invertible challenge differences when sampling without replacement in the last round [LS18]. In our framework, we handle a more general case with arbitrary k_i distinct challenges for every round i, coordinate-wise challenges, and imperfect challenge space (i.e. not every challenge difference is invertible in $\mathcal{R}_{q'}$). The last feature is in particular enabled by introducing challenge predicates. We also believe our PSS abstraction has better reusability in that it allows future protocol designers to specify arbitrary predicates for every round and then immediately derive a concrete knowledge error by invoking Theorem 4.1 without going into the details of [AFK22]. In contrast, the approach of [LNP22] is closely coupled to the extraction techniques of [AFK22] requiring a strong understanding of these previous works to apply elsewhere.

Almost Special Soundness. In [BF23] Bünz and Fisch present an extension of special-soundness (dubbed *almost special soundness (AMSS)*) which allows them to enforce additional properties. For protocols to be compatible with their approach, they require that the used commitments be deterministic and therefore non-hiding. This is part of a broader requirement that if the protocol is run again for the same challenges, a resulting accepting transcript must either be the same, or allow breaking binding. Limiting the prover responeses is core to their technique, if the prover has the freedom to choose intermediate messages without breaking binding it may significantly bias the distribution of the subsequent challenges through "grinding". While it does use deterministic commitments, LaBRADOR allows multiple prover responses when projecting the witness, to reduce the completness error the prover is given a choice of λ different projections.

B Additional Preliminaries

B.1 Signatures

We recall the standard syntax for digital signature schemes.

Definition B.1 (S). *A* signature *scheme* (S) *for a message space M consists of a tuple of* PPT *algorithm* $\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *defined as follows:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: On input the security parameter λ, the setup algorithm outputs the public parameters* pp.

$\mathsf{Gen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$*: On input the public parameters* pp*, the key generation algorithm outputs a pair of secret key* sk *and public key* pk.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$*: On input a secret key* sk *and a message $m \in M$, the signing algorithm outputs a signature σ.*

$\mathsf{Ver}(\mathsf{pk}, m, \sigma) \to b$*: On input a public key* pk*, a message $m \in M$ and a signature σ, the verification algorithm outputs either 1 (accept) or 0 (reject).*

Game 1: $\mathsf{EU\text{-}CMA}_\mathsf{S}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
$\mathcal{Q} := \varnothing$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{OSign}}(\mathsf{pp}, \mathsf{pk})$
if $\mathsf{Ver}(\mathsf{pk}, m, \sigma) \ \wedge \ m \notin \mathcal{Q}$ **then**
    return 1
else
    return 0

$\underline{\mathsf{OSign}(m)}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return σ

Definition B.2 (Correctness). *Let* $\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme for a message space* M*. It is called* correct *if for all* $\lambda \in \mathbb{N}$ *and all* $m \in M$ *it yields*

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}, m, \sigma) = 1\right] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$ *and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*.*

Definition B.3 (Unforgeability). *Let* $\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme for a message space* M*. It satisfies* existential unforgeability under adaptive chosen-message attacks ($\mathsf{EU\text{-}CMA}$) *if for all* PPT *adversaries* $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{EU\text{-}CMA}}_{\mathsf{S}}(\mathcal{A}) := \Pr\left[\mathsf{EU\text{-}CMA}_\mathsf{S}(\mathcal{A}, \lambda) = 1\right] = \mathsf{negl}(\lambda),$$

where the $\mathsf{EU\text{-}CMA}_\mathsf{S}$ *game is described in Game 1.*

A slightly stronger notion of $\mathsf{EU\text{-}CMA}$ security, needed later in Lemma C.2, allows the adversary to have a runtime that is *expected* to be polynomial time.

Definition B.4 (Unforgeability$^+$). *Let* $\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme for a message space* M*. It satisfies* special existential unforgeability under adaptive chosen-message attacks ($\mathsf{EU\text{-}CMA}^+$) *if for all probabilistic adversaries* $\mathcal{A}$ *who run in* expected *polynomial time it yields*

$$\mathsf{Adv}^{\mathsf{EU\text{-}CMA}^+}_{\mathsf{S}}(\mathcal{A}) := \Pr\left[\mathsf{EU\text{-}CMA}_\mathsf{S}(\mathcal{A}, \lambda) = 1\right] = \mathsf{negl}(\lambda),$$

where the $\mathsf{EU\text{-}CMA}_\mathsf{S}$ *game is described in Game 1.*

There is a (non-tight) generic transformation from $\mathsf{EU\text{-}CMA}$ to $\mathsf{EU\text{-}CMA}^+$ as we sketch in the following: Let $\mathcal{A}_E$ be an adversary against $\mathsf{EU\text{-}CMA}^+$ security running in expected polynomial time t_E with advantage ε_E. We assume that both values are known. We can generically construct an adversary $\mathcal{A}_W$ against $\mathsf{EU\text{-}CMA}$ running in worst-case polynomial time as follows: The adversary $\mathcal{A}_W$ runs $\mathcal{A}_E$ as a subroutine for time $2t_E/\varepsilon_E$. After this time, they stop. If $\mathcal{A}_E$ has output something, $\mathcal{A}_W$ forwards this output. Else, they output $\perp$. The runtime of $\mathcal{A}_W$ is $t_W = 2t_E/\varepsilon_E$, which is polynomial. By the Markov inequality, the advantage of $\mathcal{A}_W$ is $\varepsilon_W \geq \varepsilon_E - t_E/(2t_E/\varepsilon_E) = \varepsilon_E/2$.

B.2 Aggregate Signatures

Aggregate signatures (AS) were first introduced by Boneh et al. [BGLS03]. We recall now the syntax of an AS scheme, together with the definitions of correctness and security.

Definition B.5 (AS). *An* aggregate signature *scheme* (AS) *for a message space* M *consists of a tuple of* PPT *algorithms* $\mathsf{AS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{AggSign}, \mathsf{AggVer})$ *defined as follows:*

$\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is a signature scheme as in Definition B.1.*

$\mathsf{AggSign}(\mathsf{pp}, \{\mathsf{pk}_i, m_i, \sigma_i\}_{i \in [N]}) \to \sigma_{\mathsf{agg}}$*: On input a list of* N *message, public key and signature tuples, the aggregate signing algorithm outputs an aggregate signature* σ_{agg}*.*

$\mathsf{AggVer}(\mathsf{pp}, \{\mathsf{pk}_i, m_i\}_{i \in [N]}, \sigma_{\mathsf{agg}}) \to b$*: On input a list of* N *message, public-key tuples and an aggregate signature* σ_{agg}*, the aggregate verification algorithm either outputs* 1 *(accept) or* 0 *(reject).*

Game 2: $\mathsf{EU\text{-}ACK}_{\mathsf{AS}}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
$\mathcal{Q} := \varnothing$
$(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) \leftarrow \mathcal{A}^{\mathsf{OSign}}(\mathsf{pp}, \mathsf{pk})$
if $\mathsf{AggVer}(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) \wedge \exists i^* \in [N] : (\mathsf{pk}_{i^*} = \mathsf{pk} \wedge m_{i^*} \notin \mathcal{Q})$ **then**
  return 1
else
  return 0

$\mathsf{OSign}(m)$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return σ

Definition B.6 (Aggregate Correctness). *Let* $\mathsf{AS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{AggSign}, \mathsf{AggVer})$ *be an aggregate signature scheme for a message space* M*. It is called* correct *if for all* $\lambda, N \in \mathbb{N}$ *it yields*

$$\Pr\left[\mathsf{AggVer}(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) = 1\right] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $m_i \in M$*,* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Gen}(\mathsf{pp})$*,* $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m_i)$ *for all* $i \in [N]$ *and* $\sigma_{\mathsf{agg}} \leftarrow \mathsf{AggSign}(\{\mathsf{pk}_i, m_i, \sigma_i\}_{i\in[N]})$*.*

Definition B.7 (Aggregate Unforgeability). *An* AS *scheme satisfies* existential unforgeabilty in the aggregate chosen key model (EU-ACK)*, if for all* PPT *adversaries* $\mathcal{A}$*,*

$$\mathsf{Adv}_{\mathsf{AS}}^{\mathsf{EU\text{-}ACK}}(\mathcal{A}) := \Pr\left[\mathsf{EU\text{-}ACK}_{\mathsf{AS}}(\mathcal{A}, \lambda) = 1\right] = \mathsf{negl}(\lambda),$$

where the $\mathsf{EU\text{-}ACK}_{\mathsf{AS}}$ *game is described in Game 2.*

B.3 Falcon Signature Scheme

In the following, we describe the Falcon signature scheme from a high level perspective, focusing on the aspects that are relevant to our work. Falcon [PFH+22] is an instantiation of the GPV framework [GPV08] for lattice-based hash-then-sign signature schemes over the NTRU [HPS98] class of structured lattices.[10] It is provably secure in both the classical and quantum random oracle models [GPV08, BDF+11].

Falcon works over a power-of-two cyclotomic ring $\mathcal{R}$ modulo q. To sign some target $\boldsymbol{t} \in \mathcal{R}_q$, the signer uses their secret key together with a trapdoor preimage sampler to sample a pair $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ from a discrete Gaussian distribution such that

$$\boldsymbol{s}_1 + \boldsymbol{h}\boldsymbol{s}_2 = \boldsymbol{t} \text{ and } \|(\boldsymbol{s}_1, \boldsymbol{s}_2)\|_2 \leq \beta,$$

where $\boldsymbol{h} \in \mathcal{R}_q$ is the public key sampled from the NTRU distribution. To compress the signature, one observes that $\boldsymbol{s}_1$ can be recomputed from $\boldsymbol{s}_2$ and $\boldsymbol{t}$, so it does not need to be stored.

To sign a message $m \in \{0,1\}^*$, the target is chosen according to some hash function $\mathsf{H} : \{0,1\}^* \to \mathcal{R}_q$. However, the security of any GPV signature scheme critically relies on the signer not providing two different signatures for the same target. To prevent such collisions, three countermeasures are proposed in [GPV08]; making the signature scheme stateful, using a deterministic preimage sampler or adding a random salt. Falcon opted for the latter, sampling some salt $r \xleftarrow{\$} \{0,1\}^k$ and prepending it to the message before hashing, i.e., $\boldsymbol{t} = \mathsf{H}(r, m)$. The salt always has to be included in the signature. We call $(r, \boldsymbol{s}_1, \boldsymbol{s}_2)$ an *uncompressed* Falcon signature and $(r, \boldsymbol{s}_2)$ a *compressed* Falcon signature.

B.4 SNARKs

Definition B.8 (Binary Relation). *A* binary relation R_{pp}*, parameterized* pp*, is a set of tuples* (x, w) *where* x *is called the* statement *and* w *is the* witness*.*

Definition B.9 (Non-Interactive Argument in the ROM). *Let* R_{pp} *be a binary relation and* H *be a random oracle. A* non-interactive argument system *in the random oracle model for* R_{pp} *is a tuple of* PPT *algorithms* $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *defined as follows:*

[10] We formally recall the definition of a hash-then-sign signature scheme in Section C.1.

Table 2: Relevant Falcon parameters from [PFH+22], where λ is the security level, d the ring degree, q the modulus and k the salt bitlength. The signature size is for compressed signatures $(r, \boldsymbol{s}_2)$. The two security levels are referred to as Falcon-512 and Falcon-1024.

λ	d	q	k	Signature bytelength	Norm bound $\lfloor \beta^2 \rfloor$
128	512	12 289	320	666	34 034 726
256	1024			1280	70 265 242

$\mathcal{G}(1^\lambda) \rightarrow \mathsf{pp}$*: On input the security parameter* λ*, the setup algorithm outputs the public parameters* pp*.*

$\mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathsf{x}, \mathsf{w}) \rightarrow \pi$*: On input the public parameter* pp*, a statement* x *with a witness* w*, the prove algorithm outputs a proof* π*.*

$\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathsf{x}, \pi) \rightarrow b$*: On input the public parameter* pp*, a statement* x *and a proof* π*, the verification algorithm outputs either* 1 *(accept) or* 0 *(reject).*

We recall standard properties of a non-interactive argument system.

Definition B.10 (Completeness). *A non-interactive argument Π is called* complete *if for all* $\lambda, N \in \mathbb{N}$*, for all* $\mathsf{pp} \in \mathcal{G}(1^\lambda)$*, and for all* $(\mathsf{x}, \mathsf{w}) \in R_{\mathsf{pp}}$*, it yields*

$$\Pr\left[\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathsf{x}, \pi) = 1 \;:\; \pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathsf{x}, \mathsf{w})\right] = 1 - \mathsf{negl}(\lambda).$$

When analyzing knowledge soundness of Fiat-Shamir-transformed predicate-special-sound protocols, we rely on the following formulation.

Definition B.11 (Knowledge soundness [AFK22, Definition 9]). *A non-interactive random oracle proof* $(\mathcal{P}, \mathcal{V})$ *for a relation R is adaptively knowledge sound with knowledge error* $\kappa : \mathbb{N} \times \mathbb{N} \rightarrow [0, 1]$ *if there exists a positive polynomial p and an algorithm $\mathcal{E}$, called a knowledge extractor, with the following properties: The extractor, given input* $n \in \mathbb{N}$ *and oracle access to any adaptive Q-query random oracle prover* $\mathcal{P}^*$ *that outputs statements x with* $|x| = n$*, runs in an expected number of steps that is polynomial in n and Q (where invocations of* $\mathcal{P}^*$ *have unit cost) and outputs a tuple* $(x, \pi, \mathsf{aux}, v, w)$ *such that* $\{(x, \pi, \mathsf{aux}, v) : (x, \pi, \mathsf{aux}) \leftarrow \mathcal{P}^{*\mathsf{RO}} \wedge v \leftarrow \mathcal{V}^{\mathsf{RO}}(x, \pi)\}$ *and* $\{(x, \pi, \mathsf{aux}, v) : (x, \pi, \mathsf{aux}, v, w) \leftarrow \mathcal{E}^{\mathcal{P}^*}(n)\}$ *are identically distributed and*

$$\Pr\left[v = 1 \wedge (x, w) \in R \;\middle|\; (x, \pi, \mathsf{aux}, v, w) \leftarrow \mathcal{E}^{\mathcal{P}^*}(n)\right] \geq \frac{\varepsilon(\mathcal{P}^*) - \kappa(n, Q)}{p(n)},$$

where $\varepsilon(\mathcal{P}^*) = \Pr\left[\mathcal{V}^{\mathsf{RO}}(x, \pi) = 1 \;\middle|\; (x, \pi) \leftarrow \mathcal{P}^{*\mathsf{RO}}\right]$*. Here, $\mathcal{E}$ implements* RO *for* $\mathcal{P}^*$*. In particular, $\mathcal{E}$ can arbitrarily program* RO*. Moreover, the randomness is over the randomness of* $\mathcal{E}$*,* $\mathcal{V}$*,* $\mathcal{P}^*$ *and* RO*.*

We simply say it has *knowledge soundness* if $\kappa(n, Q) = \mathsf{negl}(\lambda)$.

Remark B.1. Note that to show this property, by the linearity of expectation, it is enough to consider deterministic provers $\mathcal{P}^*$ [AFK22, Remark 2].

To prove security of our aggregate signature scheme, we require that our argument of knowledge is still sound for provers receiving auxiliary input. Following the approach of [FN16], we provide a slightly stronger version of knowledge soundness taking this auxiliary input into account. Our formulation is tailored to random oracle-based arguments unlike the CRS-based definition of [FN16, Definition 4]. Accordingly, we make sure that auxiliary input does not depend on the public parameters and the random oracle H such that it does not interfere with rewinding-based knowledge extraction. We use the definition below to show security of SNARK-based generic construction of aggregate signatures. Additionally, we make the generation of public parameters explicit, to show that the auxiliary is produced independently of the public parameters.

Definition B.12 ($\mathcal{Z}$-auxiliary Knowledge Soundness). *We extend the knowledge soundness definition to allow the prover to receive auxiliary information. Recall $\mathcal{G}$ produces public parameters* pp*. Let $\mathcal{Z}$ be a* PPT *algorithm taking the security parameter as input and outputting some auxiliary information* $\mathsf{aux\text{-}in}$*. We make the following changes from Definition B.11, the prover is given additional inputs* $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$

and $\mathsf{aux\text{-}in} \leftarrow \mathcal{Z}(1^\lambda)$*, the extractor also receives* pp *and* $\mathsf{aux\text{-}in}$*. Knowledge soundness is now quantified over executions of* $\mathcal{G}$ *and* $\mathcal{Z}$*,*

$$\Pr\left[v = 1 \wedge (x, w) \in R_{\mathsf{pp}} \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \mathsf{aux\text{-}in} \leftarrow \mathcal{Z}(1^\lambda), \\ (x, \pi, \mathsf{aux\text{-}out}, v, w) \leftarrow \mathcal{E}^{\mathcal{P}^*(\mathsf{pp},\mathsf{aux\text{-}in})}(n, \mathsf{pp}, \mathsf{aux\text{-}in}) \end{array}\right] \geq \varepsilon(\mathcal{P}^*) - \kappa(n, Q).$$

It is relatively straightforward to check that LaBRADOR also satisfies Definition B.12 as we remark after Theorem I.1.

We say that Π is a *Proof of Knowledge* if it satisfies completeness and knowledge soundness. If knowledge soundness only holds for PPT adversaries, we say that is an *Argument* of Knowledge. Moreover, if $|\pi| \in O(\mathsf{poly}(\lambda) \cdot \mathsf{polylog}(|\mathsf{x}| + |\mathsf{w}|))$ then we say Π is *succinct*. A succinct non-interactive argument of knowledge is called a *SNARK*. A common approach for designing non-interactive protocols is transforming interactive protocols to a corresponding non-interactive protocol in the random oracle model.

Definition B.13 (Fiat-Shamir Transform). *The adaptive Fiat-Shamir transformation* $\mathsf{FS}[\Pi]$ *of a protocol* $\Pi = (\mathcal{P}, \mathcal{V})$ *is a non-interactive argument in the ROM (Definition B.9). For a statement* x *and witness* w*, the prover* $\mathcal{P}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu}$ *runs* $\mathcal{P}_\Pi$*, but rather than interacting with the verifier to obtain the challenge it instead queries the random oracle, more precisely in round* i *with prover message* a_i

$$c_i \leftarrow \mathsf{RO}_i(x, i, c_{i-1}, a_i).$$

The prover outputs proof $\pi = (a_1, \ldots, a_{\mu+1})$*. Given a proof* π*, the verifier* $\mathcal{V}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu}$ *recomputes the challenges as* $c_i \leftarrow \mathsf{RO}_i(x, i, c_{i-1}, a_i)$ *and accepts iff* $\mathcal{V}_\Pi$ *accepts the transcript* $(a_1, c_1, \ldots, c_\mu a_{\mu+1})$*.*

B.5 Special Soundness

In this section, we recap the notion of special soundness and its generalizations.

Definition B.14 (k-special-soundness). *Let* $k \in \mathbb{N}$ *and let* Π *be a 3-message public coin proof/argument of knowledge for a binary relation* R_{pp}*. We say that* Π *is* k*-special sound if there exists a polynomial time algorithm which on input a statement* x *and* k *accepting transcripts* $(a, c_1, z_1), \ldots, (a, c_k, z_k)$ *with the same first prover message* a_1 *and pairwise distinct verifier challenges* $c_1, \ldots, c_k \in \mathcal{C}$ *outputs a witness* w *such that* $(x, w) \in R$*.*

This definition can be generalized to the multi-round setting. The notion of having k accepting transcripts with same first message and distinct challenges generalizes to having an accepting $(k_1, \ldots, k_\mu)$-tree of transcripts. A single transcript is a $(1, \ldots, 1)$-tree of transcripts. Given k_μ transcripts that have the same prefix $(a_1, c_1, a_2, c_2 \ldots, a_\mu)$ and pairwise distinct μ-th challenges, we obtain a $(1, \ldots, 1, k_\mu)$-tree of transcripts. We refer to the shared prefix as their trunk. Given $k_{\mu-1}$ $(1, \ldots, 1, k_\mu)$-trees of transcripts with the same trunk $(a_1, c_1, \ldots, a_{\mu-1})$ and pairwise distinct $(\mu - 1)$-th challenges, we obtain a $(1, \ldots, 1, k_{\mu-1}, k_\mu)$-tree of transcripts, and so on. Collecting $K = \prod_{i=1}^{\mu} k_i$ transcripts in this manner, we obtain a $(k_1, \ldots, k_\mu)$-tree of transcripts. For a visualization of this, see Figure 1.

Definition B.15 ($\boldsymbol{K}$-special-soundness). *Let* $\mu, k_1, \ldots, k_\mu \in \mathbb{N}$ *and let* Π *be a* $2\mu + 1$*-message public-coin proof/argument of knowledge for a binary relation* R*. Let* $\boldsymbol{K} = (k_1, \ldots, k_\mu)$*. We say that* Π *is* $\boldsymbol{K}$*-special sound if there exists a polynomial time algorithm which on input a statement* x *and an accepting* $\boldsymbol{K}$*-tree of transcripts outputs a witness* w *such that* $(x, w) \in R$*.*

Another generalization of k-special soundness is the notion of coordinate-wise special-soundness introduced in [FMN23]. Assume that the challenge sent by the verifier is a vector $\vec{c} \in S^r$ for some set S. For each $i \in [r]$, we define the relation $\equiv_i$ for challenge vectors that only differ in the i-th coordinate. For $\vec{x} = (x_1, \ldots, x_r) \in S^r$ and $\vec{y} = (y_1, \ldots, y_r) \in S^r$,

$$\vec{x} \equiv_i \vec{y} \iff x_i \neq y_i, \forall j \in [r] \setminus \{i\} : \; x_j = y_j.$$

Using this relation, the set of permissible sets of challenge vectors is defined as

$$\mathsf{SS}(S, r, k) = \left\{ \{\vec{c}_1, \ldots, \vec{c}_K\} \subseteq 2^{(S^r)} \;\middle|\; \begin{array}{c} \exists e \in [K], \forall i \in [r], \\ \exists J = \{j_1, \ldots, j_{k-1}\} \subseteq [K] \setminus \{e\}, \\ \forall j, j' \in J \cup \{e\}, j \neq j' : \; \vec{c}_j \equiv_i \vec{c}_{j'} \end{array} \right\},$$

where $K = r(k - 1) + 1$. The vector $\vec{c}_e$ is the unique element such that any other vector in the set differs to it in exactly one coordinate. We say that $\vec{c}_e$ is the central vector. For each coordinate i, there is exactly $k - 1$ distinct vectors that differ with $\vec{c}_e$ in only the i-th coordinate. Hence, for each coordinate we have k distinct challenges.

Definition B.16 (r-coordinate-wise k-special-soundness). *Let $k, r \in \mathbb{N}$ and let Π be a 3-message public-coin proof/argument of knowledge for a relation R. Assume the challenge set of Π is S^r for some set S. We say that Π is r-coordinate-wise k-special-sound if there exists a polynomial time algorithm which on input a statement x and $r(k-1)+1$ transcripts with the same first message and challenges in $\mathsf{SS}(S, r, k)$ outputs a witness w such that $(x, w) \in R$.*

Setting $r = 1$ yields regular k-special-soundness. The notion generalizes to $2\mu+1$-message $(r_1, \ldots, r_\mu)$-coordinate-wise $(k_1, \ldots, k_\mu)$-special sound in the same way as before. Assume the m-th challenge set is $S_m^{r_m}$. Given $r_\mu(k_\mu - 1)+1$ transcripts with the same prefix $(a_1, c_1, a_2, c_2, \ldots, a_\mu)$ and μ-th challenge vectors in $\mathsf{SS}(S_\mu, r_\mu, k_\mu)$, we obtain a $(r_1, \ldots, r_\mu)$-coordinate-wise $(1, \ldots, 1, k_\mu)$-special-sound tree of transcripts. Finding $r_{\mu-1}(k_{\mu-1} - 1) + 1$ such trees with the same prefix $(a_1, c_1, \ldots, a_{\mu-1})$ and $(\mu-1)$-th challenges in $\mathsf{SS}(S_{\mu-1}, r_{\mu-1}, k_{\mu-1})$, we obtain a $(r_1, \ldots, r_\mu)$-coordinate-wise $(1, \ldots, 1, k_{\mu-1}, k_\mu)$-tree of transcripts, etc.

Definition B.17 ($\boldsymbol{R}$-coordinate-wise $\boldsymbol{K}$-special-soundness). *Let $\mu, k_1, \ldots, k_\mu, r_1, \ldots, r_\mu \in \mathbb{N}$ and let Π be a $2\mu+1$-message public-coin proof/argument of knowledge for a relation R. Let $\boldsymbol{K} = (k_1, \ldots, k_\mu)$ and let $\boldsymbol{R} = (r_1, \ldots, r_\mu)$. Assume the m-th challenge set is $S_m^{r_m}$ for some set S_m. We say that Π is $\boldsymbol{R}$-coordinate-wise $\boldsymbol{K}$-special-sound if there exists a polynomial time algorithm which on input a statement x and a $\boldsymbol{R}$-coordinate-wise $\boldsymbol{K}$-tree of transcripts outputs a witness w such that $(x, w) \in R$.*

B.6 Details of LaBRADOR

B.6.1 Module-SIS problem The security of LaBRADOR [BS23] relies on the hardness of the Module Shortest Integer Solution problem, which was first introduced in [LS15].

Definition B.18. *Let $n, m, \beta \in \mathbb{N}$. The Module Short Integer Solution problem $\mathsf{M\text{-}SIS}_{n,m,\beta}$ over $\mathcal{R}_q$ is defined as follows. Given $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m}$, find an $\vec{\boldsymbol{x}} \in \mathcal{R}_q^m$ such that $\boldsymbol{A}\vec{\boldsymbol{x}} = \vec{\boldsymbol{0}}$ and $0 < \|\vec{\boldsymbol{x}}\|_2 \leq \beta$.*

The $\mathsf{M\text{-}SIS}$ assumption states that no PPT adversary can solve this problem with non-negligible advantage. A classical worst-case to average-case reduction was provided in [LS15], showing that the $\mathsf{M\text{-}SIS}$ problem is at least as hard as finding a short basis in module lattices. The best known attacks for $\mathsf{M\text{-}SIS}_{n,m,\beta}$ do not significantly depend on m [MR09], so it is often omitted. The number n is called the module rank of the $\mathsf{M\text{-}SIS}$ instance.

B.6.2 The LaBRADOR Protocol The LaBRADOR protocol is an iterative multi-round public-coin interactive proof, which can be made non-interactive in the random oracle model by applying the Fiat-Shamir transform. Informally, each iteration takes as input a statement x and witness w and produces a transcript that is a proof of knowledge for $(\mathsf{x}, \mathsf{w}) \in R$. All of the messages in this transcript except the last message define a part of small size included in the final proof. The last message w' is sent by the prover and defines a new statement x', such that $(\mathsf{x}', \mathsf{w}') \in R$. The new witness w' is shorter than w, but might still not be very short. Instead of including the last message w' in the final proof, we prove that $(\mathsf{x}', \mathsf{w}') \in R$. Again, all but the last message are inserted in the final proof. This iterative process is repeated until we no longer make progress on the length of the final message w', at which point we output w' as the last part of the final proof.

Each iteration consists of 5 steps. Like the original paper, we present the interactive version of the protocol. Later in this paper, we concretely analyze the Fiat-Shamir transform of this protocol. For completeness, we describe the complete procedures for the first iteration of LaBRADOR prover and verifier in Protocol 2 and Protocol 3, respectively. In what follows, we explain every step in detail.

Step 1. Commit: The prover commits to each witness vector with an Ajtai commitment $\vec{\boldsymbol{v}}_i = \boldsymbol{A}\vec{\boldsymbol{w}}_i \in \mathcal{R}_q^\kappa$. Note that the norm check on the sum of the witnesses implies that $\|\vec{\boldsymbol{w}}_i\|_2 \leq \beta$. Thus, the parameter κ is set such that $\mathsf{M\text{-}SIS}_{\kappa,2\beta}$ is hard, so that the commitments are binding. Notice that for the purpose of succinctness, we do not need that the commitments are hiding.

Sending each commitment to the verifier would be costly, so instead the prover produces a single commitment $\vec{\boldsymbol{u}}_1$ to all the $\vec{\boldsymbol{v}}_i$. We refer to the $\vec{\boldsymbol{v}}_i$ as the inner commitments and $\vec{\boldsymbol{u}}_1$ as the outer commitment. Because the inner commitments are not short, they first have to be decomposed into $t_1 \geq 2$ parts with respect to a small base b_1, i.e., $\vec{\boldsymbol{v}}_i = \vec{\boldsymbol{v}}_i^{(0)} + b_1\vec{\boldsymbol{v}}_i^{(1)} + \ldots b_1^{t_1-1}\vec{\boldsymbol{v}}_i^{(t_1-1)}$. All of these parts are concatenated to $\vec{\boldsymbol{v}} \in \mathcal{R}_q^{t_1\kappa r}$. Finally, the prover commits to $\vec{\boldsymbol{v}}$ (and a vector $\vec{\boldsymbol{g}}$ that will be explained later) with $\vec{\boldsymbol{u}}_1 = \boldsymbol{B}\vec{\boldsymbol{v}} + \boldsymbol{C}\vec{\boldsymbol{g}} \in \mathcal{R}_q^{\kappa_1}$ and sends it to the verifier.

Step 2. Project: To reach a security level λ, the verifier samples $\Pi_i \leftarrow \mathsf{C}^{2\lambda\times(nd)}$ for $i = 1,\ldots,r$ and sends them to the prover. The prover responds with the combined projection $\vec{p} = \sum_{i=1}^{r} \Pi_i \tau(\vec{\boldsymbol{w}}_i) \in \mathbb{Z}_q^{2\lambda}$. The verifier checks whether $\|\vec{p}\|_2 \leq \sqrt{\lambda}\beta$. If not, the procedure is repeated until the prover can come up with a short enough $\vec{p}$.

To enforce that the final projection was computed correctly, new constraints are added. Let $\vec{\boldsymbol{\pi}}_i^{(j)}$ be the (unique) ring element corresponding to the j-th row of Π_i. For $j = 1,\ldots,2\lambda$, applying Equation 1 and using the linearity of $\mathsf{ct}(\cdot)$, we observe that

$$
\begin{aligned}
0 &= \sum_{i=1}^{r} \langle \tau(\vec{\boldsymbol{\pi}}_i^{(j)}), \tau(\vec{\boldsymbol{w}}_i)\rangle - p_j = \sum_{i=1}^{r} \mathsf{ct}\left(\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(j)}), \vec{\boldsymbol{w}}_i\rangle\right) - p_j \\
&= \mathsf{ct}\left(\sum_{i=1}^{r} \langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(j)}), \vec{\boldsymbol{w}}_i\rangle - p_j\right),
\end{aligned}
$$

where p_j is the j-th coordinate of $\vec{p}$. This defines a LaBRADOR compatible constant-term constraint.

Step 3. Aggregate constraints: The goal in this step is to aggregate all the dot product constraints in $\mathcal{F}$ and $\mathcal{F}'$ to a single dot product constraint F. First, for security level λ, $\mathcal{F}'$ is aggregated to $\lceil\lambda/\log_2(q)\rceil$ constant term constraints by taking random linear combinations. For $k = 1,\ldots,\lceil\lambda/\log_2(q)\rceil$, the verifier sends $\vec{\psi}^{(k)} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{F}'|}$, $\vec{\omega}^{(k)} \xleftarrow{\$} \mathbb{Z}_q^{2\lambda}$, which define the constraint

$$
\begin{aligned}
f''^{(k)}(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) &= \sum_{j=1}^{|\mathcal{F}'|} \psi_j^{(k)} f'^{(j)}(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) \\
&\quad + \sum_{j=1}^{2\lambda} \omega_j^{(k)} \left(\sum_{i=1}^{r} \langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(j)}), \vec{\boldsymbol{w}}_i\rangle - p_j\right) \\
&= \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j}''^{(k)} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j\rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i''^{(k)}, \vec{\boldsymbol{w}}_i\rangle - b_0''^{(k)},
\end{aligned}
$$

where $\boldsymbol{a}_{i,j}''^{(k)}, \vec{\boldsymbol{\varphi}}_i''^{(k)}$ and $b_0''^{(k)}$ are set accordingly. Next, the $f''^{(k)}$ are extended to full constraints of the type in $\mathcal{F}$. The prover extends the integers $b_0''^{(k)}$ to full polynomials $\boldsymbol{b}''^{(k)}$ such that $f''^{(k)}(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) = \boldsymbol{0}$. The prover then sends the $\boldsymbol{b}''^{(k)}$ to the verifier, which checks that the constant term of each has the correct value.

Finally, the functions in $\mathcal{F}$ and the $f''^{(k)}$ are aggregated to a single dot product constraint F by again taking a random linear combination. The verifier sends $\vec{\boldsymbol{\alpha}} \xleftarrow{\$} \mathcal{R}_q^{|\mathcal{F}|}$ and $\vec{\boldsymbol{\beta}} \xleftarrow{\$} \mathcal{R}_q^{\lceil\lambda/\log_2(q)\rceil}$, which define

$$
\begin{aligned}
F(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) &= \sum_{k=1}^{|\mathcal{F}|} \boldsymbol{\alpha}_k f^{(k)}(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) + \sum_{k=1}^{\lceil\lambda/\log_2(q)\rceil} \boldsymbol{\beta}_k f''^{(k)}(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) \\
&= \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j\rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_i\rangle - \boldsymbol{b},
\end{aligned}
$$

where again $\boldsymbol{a}_{i,j}, \vec{\boldsymbol{\varphi}}_i$ and $\boldsymbol{b}$ are set accordingly. Additionally, the verifier commits to a vector $\vec{\boldsymbol{h}}$ (that will be explained later) with $\vec{\boldsymbol{u}}_2 = \boldsymbol{D}\vec{\boldsymbol{h}}$.

Step 4. Amortize witness: In order to convince the verifier that $F(\vec{\boldsymbol{w}}_1,\ldots,\vec{\boldsymbol{w}}_r) = \boldsymbol{0}$ the prover provides a random linear combination of the witness vectors. The verifier sends challenges $\boldsymbol{c}_i \leftarrow \mathcal{C}$ for $i = 1,\ldots,r$ and the prover computes $\vec{\boldsymbol{z}} = \sum_{i=1}^{r} \boldsymbol{c}_i \vec{\boldsymbol{w}}_i$. Instead of checking that F is satisfied by the witness, the verifer will check that the following 3 dot product constraints hold:

$$
(1)\ \langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{z}}\rangle = \sum_{i,j=1}^{r} \boldsymbol{g}_{i,j}\boldsymbol{c}_i\boldsymbol{c}_j, \qquad (2)\ \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{z}}\rangle \boldsymbol{c}_i = \sum_{i,j=1}^{r} \boldsymbol{h}_{i,j}\boldsymbol{c}_i\boldsymbol{c}_j,
$$

$$
(3)\ \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j}\boldsymbol{g}_{i,j} + \sum_{i=1}^{r} \boldsymbol{h}_{i,i} - \boldsymbol{b} = \boldsymbol{0}
$$

Here the $\boldsymbol{g}_{i,j}$ and the $\boldsymbol{h}_{i,j}$ are garbage polynomials, with $\boldsymbol{g}_{i,j} = \boldsymbol{g}_{j,i}$ and $\boldsymbol{h}_{i,j} = \boldsymbol{h}_{j,i}$ for $i, j \in [r]$. Due to this symmetry, instead of r^2 many, only $r(r+1)/2$ many of them need to be stored, respectively. The honest prover computes $\boldsymbol{g}_{i,j} = \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle$ as well as $\boldsymbol{h}_{i,j} = \frac{1}{2}\left(\langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_j \rangle + \langle \vec{\boldsymbol{\varphi}}_j, \vec{\boldsymbol{w}}_i \rangle\right)$. Like the $\vec{\boldsymbol{v}}_i$, the $\boldsymbol{h}_{i,j}$ are each decomposed into t_1 parts with respect to the basis b_1, and concatenated to the vector $\vec{\boldsymbol{h}} \in \mathcal{R}_q^{t_1(r^2+r)/2}$. The vector $\vec{\boldsymbol{h}}$ was the one already used in Step 3. Similarly, the $\boldsymbol{g}_{i,j}$ are decomposed into t_2 parts w.r.t. the basis b_2, and concatenated to $\vec{\boldsymbol{g}} \in \mathcal{R}_q^{t_2(r^2+r)/2}$. The vector $\vec{\boldsymbol{g}}$ is the one already used in Step 1. Finally, $\vec{\boldsymbol{z}}$ is decomposed into 2 parts $\vec{\boldsymbol{z}}_1, \vec{\boldsymbol{z}}_2$ w.r.t. the basis b.

For the final message, the prover sends $\vec{\boldsymbol{z}}_1, \vec{\boldsymbol{z}}_2, \vec{\boldsymbol{v}}, \vec{\boldsymbol{g}}, \vec{\boldsymbol{h}}$. The verifier checks the following:

1. That the above constraints (1), (2) and (3) are satisfied.
2. That $\vec{\boldsymbol{z}}$ is a somewhat short amortized opening to the $\vec{\boldsymbol{v}}_i$'s, so that $\boldsymbol{A}\vec{\boldsymbol{z}} = \sum_{i=1}^{r} \boldsymbol{c}_i \vec{\boldsymbol{v}}_i$.
3. That $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{g}})$ and $\vec{\boldsymbol{h}}$ are somewhat short openings to respectively $\vec{\boldsymbol{u}}_1$ and $\vec{\boldsymbol{u}}_2$.

<u>Step 5. Recurse:</u> Instead of including the last message in the transcript, we recursively run the protocol with it as the new witness. The three verification checks from above that the last message should satisfy are translated to a new statement in R.

First, define $\vec{\boldsymbol{e}} = \vec{\boldsymbol{v}} \| \vec{\boldsymbol{g}} \| \vec{\boldsymbol{h}} \in \mathcal{R}_q^m$ with $m = rt_1\kappa + (t_1 + t_2)(r^2 + r)/2$. The new set of witness vectors are now $\vec{\boldsymbol{z}}_1, \vec{\boldsymbol{z}}_2, \vec{\boldsymbol{e}}$. Note that they can all be made to have the rank $n' = \max(n, m)$ by padding with 0's when necessary. The verification equations depend linearly on $\vec{\boldsymbol{v}}, \vec{\boldsymbol{g}}$ and $\vec{\boldsymbol{h}}$. Hence, it is easy to see how to reformulate them to dot product constraints $\widetilde{\mathcal{F}}$ in $\vec{\boldsymbol{z}}_1, \vec{\boldsymbol{z}}_2, \vec{\boldsymbol{e}}$. The norm checks for the commitment openings are combined into one global check $\|\vec{\boldsymbol{z}}_1\|_2^2 + \|\vec{\boldsymbol{z}}_2\|_2^2 + \|\vec{\boldsymbol{e}}\|_2^2 \leq \beta'^2$. The purpose of the original norm checks was to ensure that the openings were binding. For appropriate parameters, this is still implied by the combined ℓ_2-norm bound β'. Thus, we arrive at a statement in R.

Next, the witness is folded to give better control of the parameters in the next iteration. Given folding parameters ν and μ, $\vec{\boldsymbol{z}}_1, \vec{\boldsymbol{z}}_2 \in \mathcal{R}_q^n$ are folded into ν pieces in $\mathcal{R}_q^{\lceil n/\nu \rceil}$ and $\vec{\boldsymbol{e}}$ is folded into μ pieces in $\mathcal{R}_q^{\lceil m/\mu \rceil}$. By this, we mean that we obtain $r' = 2\nu + \mu$ witness vectors $\vec{\boldsymbol{w}}'_1, \ldots, \vec{\boldsymbol{w}}'_{2\nu+\mu}$ such that $\vec{\boldsymbol{z}}_1 = \vec{\boldsymbol{w}}'_1 \| \ldots \| \vec{\boldsymbol{w}}_\nu$, $\vec{\boldsymbol{z}}_2 = \vec{\boldsymbol{w}}'_{\nu+1} \| \ldots \| \vec{\boldsymbol{w}}_{2\nu}$ and $\vec{\boldsymbol{e}} = \vec{\boldsymbol{w}}'_{2\nu+1} \| \ldots \| \vec{\boldsymbol{w}}_{2\nu+\mu}$. Padding with 0's when necessary, the new witness vectors have rank $n' = \max(\frac{n}{\nu}, \frac{m}{\mu})$. The folding parameters are chosen such that $\frac{n}{\nu} \approx \frac{m}{\mu}$, to minimize the padding required. Reformulating the constraints in terms of these new witness vectors is easy, since in general $\langle \vec{\boldsymbol{x}}_1 \| \vec{\boldsymbol{x}}_2, \vec{\boldsymbol{y}}_1 \| \vec{\boldsymbol{y}}_2 \rangle = \langle \vec{\boldsymbol{x}}_1, \vec{\boldsymbol{y}}_1 \rangle + \langle \vec{\boldsymbol{x}}_2, \vec{\boldsymbol{y}}_2 \rangle$. The same combined ℓ_2-norm bound β' still applies. Hence, $(w'_1, \ldots, w'_{r'}$ define the witness for the statement $(\widetilde{\mathcal{F}}, \beta')$ in R, which is to be used in the next iteration.

The goal of the recursion is to reduce the witness rank n. The strategy is to carefully pick ν, μ at each iteration such that enough progress is made in reducing n' while not blowing up r'. Since $m' = O(r'^2)$, too large ν, μ will be counterproductive to further reducing n'. For the recursion, Beullens and Seiler set the decomposition parameters heuristically such that $b \approx b_1 \approx b_2$, which ensures that all the witness vectors have roughly the same width (i.e., the same ℓ_∞-norm). They also want to be in the balanced state where $2n \approx m$, where the ℓ_2-norm contribution of $\vec{\boldsymbol{z}}_1 \| \vec{\boldsymbol{z}}_2$ and $\vec{\boldsymbol{e}}$ is about the same. The strategy will be to reduce $\frac{n}{\nu}$ as much as possible while maintaining that $\frac{n}{\nu} \approx \frac{m}{\mu}$ and that $2n' \approx m'$. Asymptotically, the optimal choice is $r' = O(r^{1/3})$, yielding $n' = O(r^{2/3})$. Hence, only $O(\log \log n)$ iterations of the base protocol are needed.

Protocol 2: The LaBRADOR Protocol

Relation R Consists of tuples of statement $\mathsf{x} = (((\boldsymbol{a}_{i,j}^{(k)})_{i,j\in[r]}, (\vec{\boldsymbol{\varphi}}_i^{(k)})_{i\in[r]}, \boldsymbol{b}^{(k)})_{k\in[K]}, ((\boldsymbol{a}_{i,j}'^{(l)})_{i,j\in[r]}, (\vec{\boldsymbol{\varphi}}_i'^{(l)})_{i\in[r]}, b_0'^{(l)})_{l\in[L]}, \beta)$ and witness $\mathsf{w} = (\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r)$ where $\vec{\boldsymbol{w}}_i \in \mathcal{R}_{q'}^n$ for $i \in [r]$ such that

$$f^{(k)}(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) = \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j}^{(k)} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i^{(k)}, \vec{\boldsymbol{w}}_i \rangle - \boldsymbol{b}^{(k)} = \boldsymbol{0}, \forall k \in [K]$$

$$\mathsf{ct}\left(f'^{(l)}(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r)\right) = \mathsf{ct}\left(\sum_{i,j=1}^{r} \boldsymbol{a}_{i,j}'^{(l)} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i'^{(l)}, \vec{\boldsymbol{w}}_i \rangle - b_0'^{(l)}\right) \bmod q', \forall l \in [L]$$

$$\sum_{i=1}^{r} \|\vec{\boldsymbol{w}}_i\|_2^2 \leq \beta^2.$$

$\mathcal{G}(1^\lambda)$ Uniformly sample $\boldsymbol{A} \in \mathcal{R}_{q'}^{\kappa\times n}, \boldsymbol{B}_{ik} \in \mathcal{R}_{q'}^{\kappa_1\times\kappa}$ for $i \in [1, r]$, $k \in [0, t_1 - 1]$, $\boldsymbol{C}_{ijk} \in \mathcal{R}_{q'}^{\kappa_2\times 1}$ for $i \in [1, r]$, $j \in [i, r]$, $k \in [0, t_2 - 1]$, and $\boldsymbol{D}_{i,j,k} \in \mathcal{R}_{q'}^{\kappa_2\times 1}$ for $i \in [1, r]$, $j \in [i, r]$, $k \in [0, t_1 - 1]$. Output $\mathsf{pp} = (\boldsymbol{A}, \boldsymbol{B}_{ik}, \boldsymbol{C}_{ijk}, \boldsymbol{D}_{ijk})$

$\mathcal{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w})$

1. **Commit**:
   1: For $i \in [r]$: $\vec{\boldsymbol{v}}_i = \boldsymbol{A}\vec{\boldsymbol{w}}_i = \vec{\boldsymbol{v}}_i^{(0)} + \ldots + \vec{\boldsymbol{v}}_i^{(t_1-1)} b_1^{t_1-1}$ // Generate inner commitments
   2: For $i \in [r], j \in [i, r]$: $\boldsymbol{g}_{ij} = \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle = \vec{\boldsymbol{g}}_{ij}^{(0)} + \ldots + \vec{\boldsymbol{g}}_{ij}^{(t_2-1)} b_2^{t_2-1}$
   3: $\vec{\boldsymbol{u}}_1 = \sum_{i=1}^{r} \sum_{k=0}^{t_1-1} \boldsymbol{B}_{ik} \vec{\boldsymbol{v}}_i^{(k)} + \sum_{i\leq j} \sum_{k=0}^{t_2-1} \boldsymbol{C}_{ijk} \boldsymbol{g}_{ij}^{(k)}$ // Generate outer commitment
   4: Send $\vec{\boldsymbol{u}}_1$ to $\mathcal{V}$
2. **Projection** Upon receiving $\Pi_i = (\vec{\pi}_i^{(j)})_{j\in[2\lambda]} \xleftarrow{\$} \mathsf{C}^{2\lambda\times(nd)}$ for $i \in [r]$ from $\mathcal{V}$:
   1: For $j \in [2\lambda]$: $p_j = \sum_{i=1}^{r} \langle \vec{\pi}_i^{(j)}, \vec{w}_i \rangle$
   2: Send $\vec{p} = (p_j)_{j\in[2\lambda]}$ to $\mathcal{V}$
3. **Aggregating** $f'^{(l)}$ Let $K'' = \lceil \lambda / \log q \rceil$. Upon receiving $\psi_l^{(k)}, \omega_j^{(k)} \in \mathbb{Z}_{q'}$ for $k \in [K'']$, $l \in [L]$, $j \in [2\lambda]$ from $\mathcal{V}$:
   1: For $k \in [K'']$: $\boldsymbol{a}_{ij}''^{(k)} = \sum_{l=1}^{L} \psi_l^{(k)} \boldsymbol{a}_{ij}'^{(l)}$, $\vec{\boldsymbol{\varphi}}_i''^{(k)} = \sum_{l=1}^{L} \psi_l^{(k)} \vec{\boldsymbol{\varphi}}_i^{(l)} + \sum_{j=1}^{2\lambda} \omega_j^{(k)} \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(j)})$, $b_0''^{(k)} = \sum_{l=1}^{L} \psi_l^{(k)} b_0'^{(l)} + \langle \vec{\omega}^{(k)}, \vec{p} \rangle$,

$$\begin{aligned}
f''^{(k)}(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) &:= \sum_{l=1}^{L} \psi_l^{(k)} f'^{(l)}(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) + \sum_{j=1}^{2\lambda} \omega_j^{(k)} (\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(j)}), \vec{\boldsymbol{w}}_i \rangle - p_j) \\
&= \sum_{i,j=1}^{r} \boldsymbol{a}_{ij}''^{(k)} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i''^{(k)}, \vec{\boldsymbol{w}}_i \rangle - b_0''^{(k)} \\
\boldsymbol{b}''^{(k)} &:= \sum_{i,j=1}^{r} \boldsymbol{a}_{ij}''^{(k)} \langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i''^{(k)}, \vec{\boldsymbol{w}}_i \rangle
\end{aligned}$$

   2: Send $(\boldsymbol{b}''^{(k)})_{k\in[K'']}$ to $\mathcal{V}$.
4. **Aggregating linear constraints** Upon receiving $\vec{\boldsymbol{\alpha}} \in \mathcal{R}_{q'}^{K}$ and $\vec{\boldsymbol{\beta}} \in \mathcal{R}_{q'}^{K''}$ from $\mathcal{V}$:
   1: For $i, j \in [r]$: $\vec{\boldsymbol{\varphi}}_i = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \vec{\boldsymbol{\varphi}}_i^{(k)} + \sum_{k=1}^{K''} \boldsymbol{\beta}_k \vec{\boldsymbol{\varphi}}_i''^{(k)}$
   2: For $i \in [r], j \in [i, r]$: $(\langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_j \rangle + \langle \vec{\boldsymbol{\varphi}}_j, \vec{\boldsymbol{w}}_i \rangle)/2 = \boldsymbol{h}_{ij}^{(0)} + \ldots + \boldsymbol{h}_{ij}^{(t_1-1)} b_1^{t_1-1}$
   3: Send $\vec{\boldsymbol{u}}_2 = \sum_{i\leq j} \sum_{k=0}^{t_1-1} \boldsymbol{D}_{ijk} \boldsymbol{h}_{ij}^{(k)}$ to $\mathcal{V}$
5. **Amortizing opening proof** Upon receiving $\boldsymbol{c}_i \in \mathcal{C}$ for $i \in [r]$:
   1: $\vec{\boldsymbol{z}} = \sum_{i=1}^{r} \boldsymbol{c}_i \vec{\boldsymbol{w}}_i$
   2: Send $\vec{\boldsymbol{z}}$ and $\vec{\boldsymbol{v}}_i, \boldsymbol{g}_{i,j}, \boldsymbol{h}_{ij}$ for $i \in [r], j \in [i, r]$ to $\mathcal{V}$.

Protocol 3: The LaBRADOR Protocol (cont.)

$\mathcal{V}(\mathsf{pp}, \mathsf{x}, \tau)$

1. Parse $\tau = (\vec{\boldsymbol{u}}_1, (\vec{\pi}_i^{(j)})_{i\in[r],j\in[2\lambda]}, \vec{p}, (\psi_l^{(k)}, \omega_j^{(k)})_{k\in[K''],l\in[L],j\in[2\lambda]}, (\boldsymbol{b}''^{(k)})_{k\in[K'']}, \vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{\boldsymbol{u}}_2, (\boldsymbol{c}_i)_{i\in[r]}, \vec{\boldsymbol{z}}, (\vec{\boldsymbol{v}}_i, \boldsymbol{g}_{i,j}, \boldsymbol{h}_{ij})_{i\in[r],j\in[i,r]})$
2. **Check constant term of $\boldsymbol{b}''^{(k)}$** Check $b_0''^{(k)} = \sum_{l=1}^{L} \psi_l^{(k)} b_0'^{(l)} + \langle \vec{\omega}^{(k)}, \vec{p}\rangle$
3. **Compute aggregated relation** Define aggregated statement F such that

$$\begin{aligned} F(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) &= \sum_{k=1}^{K} \boldsymbol{\alpha}_k f^{(k)}(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) + \sum_{k=1}^{K''} \boldsymbol{\beta}_k f''^{(k)}(\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r) \\ &= \sum_{i,j=1}^{r} \boldsymbol{a}_{i,j}\langle \vec{\boldsymbol{w}}_i, \vec{\boldsymbol{w}}_j\rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_i\rangle - \boldsymbol{b} = 0 \end{aligned}$$

   1: For $k \in [K'']$: $\boldsymbol{a}_{ij}''^{(k)} = \sum_{l=1}^{L} \psi_l^{(k)} \boldsymbol{a}_{ij}'^{(l)}$, $\vec{\boldsymbol{\varphi}}_i''^{(k)} = \sum_{l=1}^{L} \psi_l^{(k)} \vec{\boldsymbol{\varphi}}_i^{(l)} + \sum_{j=1}^{2\lambda} \omega_j^{(k)} \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(j)})$
   2: For $i \in [r], j \in [i,r]$: $\boldsymbol{a}_{ij} = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \boldsymbol{a}_{ij}^{(k)} + \sum_{k=1}^{K''} \boldsymbol{\beta}_k \boldsymbol{a}_{ij}''^{(k)}$
   3: For $i \in [r]$: $\vec{\boldsymbol{\varphi}}_i = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \boldsymbol{\varphi}_i^{(k)} + \sum_{k=1}^{K''} \boldsymbol{\beta}_k \boldsymbol{\varphi}_i''^{(k)}$
   4: $\boldsymbol{b} = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \boldsymbol{b}^{(k)} + \sum_{k=1}^{K''} \boldsymbol{\beta}_k \boldsymbol{b}''^{(k)}$
4. **Check amortized opening of inner commitments** Check $\boldsymbol{A}\vec{\boldsymbol{z}} = \sum_{i=1}^{r} \boldsymbol{c}_i \vec{\boldsymbol{v}}_i$
5. **Check aggregated innerproduct constraints** Check $\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{z}}\rangle = \sum_{i,j=1}^{r} \boldsymbol{g}_{ij} \boldsymbol{c}_i \boldsymbol{c}_j$
6. **Check aggregated linear constraints** Check $\sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{z}}\rangle \boldsymbol{c}_i = \sum_{i,j=1}^{r} \boldsymbol{h}_{ij} \boldsymbol{c}_i \boldsymbol{c}_j$
7. **Check aggregated relation** Check $\sum_{i,j=1}^{r} \boldsymbol{a}_{ij} \boldsymbol{g}_{ij} + \sum_{i=1}^{r} \boldsymbol{h}_{ii} - \boldsymbol{b} = 0$
8. **Check norms of decomposed inner commitments**
   1: $\vec{\boldsymbol{z}} = b\vec{\boldsymbol{z}}^{(1)} + \vec{\boldsymbol{z}}^{(0)}$
   2: For $i \in [r]$: $\vec{\boldsymbol{v}}_i = \vec{\boldsymbol{v}}_i^{(0)} + \ldots + \vec{\boldsymbol{v}}_i^{(t_1-1)} b_1^{t_1-1}$
   3: For $i \in [r], j \in [i,r]$: $\boldsymbol{g}_{ij} = \boldsymbol{g}_{ij}^{(0)} + \ldots + \boldsymbol{g}_{ij}^{(t_2-1)} b_2^{t_2-1}$
   4: For $i \in [r], j \in [i,r]$: $\boldsymbol{h}_{ij} = \boldsymbol{h}_{ij}^{(0)} + \ldots + \boldsymbol{h}_{ij}^{(t_1-1)} b_1^{t_1-1}$
   5: Check $\sum_{k=0}^{1} \left\| \vec{\boldsymbol{z}}^{(k)} \right\|^2 + \sum_{i=1}^{r} \sum_{k=0}^{t_1-1} \left\| \vec{\boldsymbol{v}}_i^{(k)} \right\|^2 + \sum_{i=1}^{r} \sum_{k=0}^{t_2-1} \left\| \boldsymbol{g}_{ij}^{(k)} \right\|^2 + \sum_{i=1}^{r} \sum_{k=0}^{t_1-1} \left\| \boldsymbol{h}_{ij}^{(k)} \right\|^2 \leq \beta'$
9. **Check opening of outer commitments** Check $\vec{\boldsymbol{u}}_1 = \sum_{i=1}^{r} \sum_{k=0}^{t_1-1} \boldsymbol{B}_{ik} \vec{\boldsymbol{v}}_i^{(k)} + \sum_{i\leq j} \sum_{k=0}^{t_2-1} \boldsymbol{C}_{ijk} \boldsymbol{g}_{ij}^{(k)}$ and $\vec{\boldsymbol{u}}_2 = \sum_{i\leq j} \sum_{k=0}^{t_1-1} \boldsymbol{D}_{ijk} \boldsymbol{h}_{ij}^{(k)}$

C From SNARKs to Aggregate Signatures

In this section, we show how to generically construct an aggregate signature from a hash-then-sign signature and a non-interactive argument system.

C.1 Hash-then-Sign Signatures

We define the class of hash-then-sign signature schemes. It is easy to see that Falcon [PFH+22] falls under the category.

Definition C.1 (HtS). *A signature scheme* $\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *for a message space* M *is said to be in the class of (randomized)* hash-then-sign HtS *with the random oracle* $\mathsf{G} : \{0,1\}^* \to \mathsf{Ra}$ *if each algorithm proceeds as follows.*

$\mathsf{Setup}(1^\lambda)$*: Outputs* pp *that defines a family of preimage sampleable functions (PSF)* $\mathcal{F} = \{F_\mathsf{k} : \mathsf{Do} \to \mathsf{Ra}\}_\mathsf{k}$.

$\mathsf{Gen}(\mathsf{pp})$*: Outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$ *which defines a PSF* $F_\mathsf{pk} \in \mathcal{F}$*. The* pk *allows computing* $y = F_\mathsf{pk}(x)$ *for* $x \in \mathsf{Do}$*, while* sk *allows sampling a preimage* x *from some distribution* $\mathscr{D}$ *defined over a set* $F_\mathsf{pk}^{-1}(y)$ *for any* $y \in \mathsf{Ra}$*. We write* $x \leftarrow \mathsf{SampleD}(\mathsf{sk}, y, \mathscr{D})$ *to denote sampling* x *from* $\mathscr{D}(F_\mathsf{pk}^{-1}(y))$*.*

$\mathsf{Sign}^\mathsf{G}(\mathsf{sk}, m)$*:*

1. *Sample a uniformly random salt* $r \in \{0,1\}^k$
2. $y = \mathsf{G}(r, m)$
3. $x \leftarrow \mathsf{SampleD}(\mathsf{sk}, y, \mathscr{D})$
4. *Output* $\sigma = (r, x)$

$\mathsf{Ver}^\mathsf{G}(\mathsf{pk}, m, \sigma)$*:*

1. $y = \mathsf{G}(r, m)$
2. *Output 1 iff* $x \in \mathsf{Do}$ *and* $F_\mathsf{pk}(x) = y$

C.2 Snarky Aggregate Hash-then-Sign Signatures

Let $\mathsf{S} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}) \in \mathsf{HtS}$ be a hash-then-sign signature scheme for a message space M and let $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a non-interactive argument system in the ROM for the binary relation $R_\mathsf{pp} = \big\{((\mathsf{pk}_i, y_i), x_i)_{i\in[N]} : \forall i \in [N], F_{\mathsf{pk}_i}(x_i) = y_i \wedge x_i \in \mathsf{Do}\big\}$. Let H be a random oracle for Π and G be a random oracle for S, respectively. We now construct an aggregate signature scheme $\mathsf{AS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{AggSign}, \mathsf{AggVer})$ as follows:

$\mathsf{Setup}(1^\lambda)$: Run $\mathsf{S}.\mathsf{Setup}(1^\lambda) \to \mathsf{pp}_\mathsf{S}$ and $\Pi.\mathcal{G}(1^\lambda) \to \mathsf{pp}_\Pi$, set $\mathsf{pp} := (\mathsf{pp}_\mathsf{S}, \mathsf{pp}_\Pi)$ and output pp.

$\mathsf{Gen}(\mathsf{pp})$: Parse $\mathsf{pp} = (\mathsf{pp}_\mathsf{S}, \mathsf{pp}_\Pi)$, run $\mathsf{S}.\mathsf{Gen}(\mathsf{pp}_\mathsf{S}) \to (\mathsf{sk}, \mathsf{pk})$, and output $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}^\mathsf{G}(\mathsf{sk}, m)$: Run $\mathsf{S}.\mathsf{Sign}^\mathsf{G}(\mathsf{sk}, m) \to \sigma = (r, x)$ and output σ.

$\mathsf{Ver}^\mathsf{G}(\mathsf{pk}, m, \sigma)$: Run $\mathsf{S}.\mathsf{Ver}^\mathsf{G}(\mathsf{pk}, m, \sigma) \to b$ and output b.

$\mathsf{AggSign}^{\mathsf{G},\mathsf{H}}(\mathsf{pp}, \{\mathsf{pk}_i, m_i, \sigma_i\}_{i\in[N]})$: Parse $\mathsf{pp} = (\mathsf{pp}_\mathsf{S}, \mathsf{pp}_\Pi)$ and $\sigma_i = (r_i, x_i)$, compute $y_i = \mathsf{G}(r_i, m_i)$ for $i \in [N]$, set $\mathsf{x} := \{\mathsf{pk}_i, y_i\}_{i\in[N]}$ and $\mathsf{w} := \{x_i\}_{i\in[N]}$, run $\Pi.\mathcal{P}^\mathsf{H}(\mathsf{pp}_\Pi, \mathsf{x}, \mathsf{w}) \to \pi$, set $\sigma_\mathsf{agg} := (\pi, (r_i)_{i\in[N]})$ and output σ_agg.

$\mathsf{AggVer}^{\mathsf{G},\mathsf{H}}(\mathsf{pp}, \{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_\mathsf{agg})$: Parse $\mathsf{pp} = (\mathsf{pp}_\mathsf{S}, \mathsf{pp}_\Pi)$ and $\sigma_\mathsf{agg} = (\pi, (r_i)_{i\in[N]})$, compute $y_i = \mathsf{G}(r_i, m_i)$ for $i \in [N]$, set $\mathsf{x} := \{\mathsf{pk}_i, y_i\}_{i\in[N]}$ and run $\Pi.\mathcal{V}^\mathsf{H}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) \to b$ and output b.

Remark C.1. In theory, the above construction does not satisfy the definition of compactness even if the proof system Π is succinct: σ_agg contains N salts individually generated by different signers. We show later in section 6 that for our parameter regimes the size of salts does not significantly impact the size of the aggregate signature.

Lemma C.1. *If* S *is correct and* Π *is complete, then* AS *is correct.*

Proof. Let $\mathsf{w} := \{x_i\}_{i\in[N]}$ be honestly generated preimages with respect to $\mathsf{x} := \{\mathsf{pk}_i, y_i\}_{i\in[N]}$, where $y_i = \mathsf{G}(r_i, m_i)$ and $\sigma_i = (r_i, x_i)$. By the correctness of S and the union bound it holds

$$\Pr\left[(\mathsf{x},\mathsf{w}) \notin R\right] = \Pr\left[\exists i \in [N] \colon \mathsf{Ver}^{\mathsf{G}}(\mathsf{pk}_i, m_i, \sigma_i) = 0\right] \leq N \cdot \mathsf{negl}(\lambda).$$

Now, by completeness of Π it holds

$$\begin{aligned}
&\Pr\left[\mathsf{AggVer}^{\mathsf{G},\mathsf{H}}(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) = 0\right] \\
=&\Pr\left[\Pi.\mathsf{Ver}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 0\right] \\
=&\Pr\left[\Pi.\mathsf{Ver}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 0 \wedge (\mathsf{x},\mathsf{w}) \in R\right] + \Pr\left[\Pi.\mathsf{Ver}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 0 \wedge (\mathsf{x},\mathsf{w}) \notin R\right] \\
\leq&\Pr\left[\Pi.\mathsf{Ver}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 0 \wedge (\mathsf{x},\mathsf{w}) \in R\right] + \Pr\left[(\mathsf{x},\mathsf{w}) \notin R\right] \\
\leq&\Pr\left[\Pi.\mathsf{Ver}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 0 | (\mathsf{x},\mathsf{w}) \in R\right] + \Pr\left[(\mathsf{x},\mathsf{w}) \notin R\right] \\
\leq& \mathsf{negl}(\lambda) + N \cdot \mathsf{negl}(\lambda).
\end{aligned}$$

Hence, for all $\lambda, N \in \mathbb{N}$ it yields

$$\Pr\left[\mathsf{AggVer}^{\mathsf{G},\mathsf{H}}(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) = 1\right] = 1 - \mathsf{negl}(\lambda).$$

□

Lemma C.2. *If* S *is* $\mathsf{EU\text{-}CMA}^+$ *secure,* $k \in \mathsf{poly}(\lambda)$*, and* Π *is* $\mathcal{Z}$*-auxiliary input knowledge sound, then* AS *is* $\mathsf{EU\text{-}ACK}$ *secure in the random oracle model, where* $\mathcal{Z}(1^\lambda)$ *proceeds as follows: run* $\mathsf{pp}_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{S.Gen}(\mathsf{pp}_{\mathsf{S}})$*,* $y_i \xleftarrow{\$} \mathsf{Ra}$*,* $x_i \leftarrow \mathsf{SampleD}(\mathsf{sk}, y_i, \mathscr{D})$ *for* $i = 1, \ldots, Q_s$*,* $\bar{y}_i \xleftarrow{\$} \mathsf{Ra}$ *for* $i = 1, \ldots, Q_g$ *and output* $(\mathsf{pp}_{\mathsf{S}}, \mathsf{pk}, x_1, y_1, \ldots, x_{Q_s}, y_{Q_s}, \bar{y}_1, \ldots, \bar{y}_{Q_g})$*, where* Q_s *is the number of signing queries and* Q_g *is the number queries to the random oracle* G*, made by an* $\mathsf{EU\text{-}ACK}$ *adversary respectively. Let* Q_h *be the number of queries to the random oracle* H*.*

Proof. We show that if there exists an adversary $\mathcal{A}$ breaking the $\mathsf{EU\text{-}ACK}$ security of AS with non-negligible probability, while assuming that Π has a negligible knowledge error κ, then we can build an adversary $\mathcal{B}$, who uses the extractor of Π, to break the $\mathsf{EU\text{-}CMA}^+$ security of S. The proof strategy below closely follows Theorem 4 of [FN16] adapted to the setting of aggregate signatures.

Our reduction $\mathcal{B}$ essentially proceeds in three steps. First, upon receiving the challenge public key pk from the $\mathsf{EU\text{-}CMA}^+$ security game, the reduction $\mathcal{B}$ queries OSign with a fixed message m to obtain a sequence of signatures and then queries G with another fixed message $\bar{m} \neq m$ and distinct salts $\bar{r}_i$ as inputs to obtain hashes $\bar{y}_i$, respectively. Second, $\mathcal{B}$ defines a cheating SNARK prover $\mathcal{P}$ which internally runs $\mathcal{A}$ and simulates its view in the $\mathsf{EU\text{-}ACK}$ game using signatures and random hash outputs as auxiliary information. Finally, $\mathcal{B}$ runs a knowledge extractor $\mathcal{E}$ against $\mathcal{P}$ to get signatures corresponding to the statement output by $\mathcal{P}$. If $\mathcal{A}$ wins the $\mathsf{EU\text{-}ACK}$ game, at least one of the signatures contains x^* such that it satisfies $F_{\mathsf{pk}}(x^*) = \bar{y}_i = \mathsf{G}(\bar{r}_i, \bar{m})$ for some i and the challenge public key pk. Such x^* indeed qualifies as a forgery breaking the $\mathsf{EU\text{-}CMA}^+$ security of S.

In more detail, the reduction $\mathcal{B}$, upon receiving $(\mathsf{pp}_{\mathsf{S}}, \mathsf{pk})$, first obtains $\bar{y}_i$ for $i = 1, \ldots, Q_g$ from G by querying G with some $(\bar{r}_i, \bar{m}) \in \{0,1\}^k \times M$, where $\bar{r}_i$ are distinct. Then $\mathcal{B}$ picks $m \neq \bar{m}$ and queries OSign Q_s times with m to retrieve signatures $(r_i, x_i)_{i\in[Q_s]}$ such that $\mathsf{G}(r_i, m) = y_i$, $F_{\mathsf{pk}}(x_i) = y_i$ and $x_i \in \mathsf{Do}$. $\mathcal{B}$ prepares pp_Π for the proof system by running $\mathsf{pp}_\Pi \leftarrow \Pi.\mathcal{G}(1^\lambda)$.

Next, consider a cheating prover $\mathcal{P}(\mathsf{pp}_\Pi, \mathsf{aux\text{-}in})$ given access to the random oracle H (whose responses are simulated by $\mathcal{B}$), where $\mathsf{aux\text{-}in} = (\mathsf{pp}_{\mathsf{S}}, \mathsf{pk}, x_1, y_1, \ldots, x_{Q_s}, y_{Q_s}, \bar{y}_1, \ldots, \bar{y}_{Q_g})$. The prover $\mathcal{P}$ proceeds as follows to simulate the view of $\mathcal{A}$ playing the $\mathsf{EU\text{-}ACK}$ game, in which the random oracles are denoted by G' and H', and the signing oracle is denoted by OSign', respectively.

1. Run $\mathcal{A}$ on input $((\mathsf{pp}_\Pi, \mathsf{pp}_{\mathsf{S}}), \mathsf{pk})$.
2. Upon receiving a query to H' from $\mathcal{A}$, relay that query to H and forward the response from H to $\mathcal{A}$.
3. Upon receiving an ith query with message m'_i to OSign' from $\mathcal{A}$, sample uniform $r'_i \in \{0,1\}^k$, and abort if the response for $\mathsf{G}'(r'_i, m'_i)$ is already defined. Else, consume (x_i, y_i) in $\mathsf{aux\text{-}in}$ to program $\mathsf{G}'(r'_i, m'_i) := y_i$ and return (r'_i, x_i) as a signature.
4. Upon receiving an ith fresh query $(\bar{r}'_i, \bar{m}'_i)$ to G' from $\mathcal{A}$, consume $\bar{y}_i$ in $\mathsf{aux\text{-}in}$ to program $\mathsf{G}'(\bar{r}'_i, \bar{m}'_i) := \bar{y}_i$ and return $\bar{y}_i$ to $\mathcal{A}$.

5. Upon receiving a forgery $(\mathsf{pk}_i^*, m_i^*)_{i\in[N]}$ and $(\pi, (r_i^*)_{i\in[N]})$, let $y_i^* = \mathsf{G}'(r_i^*, m_i^*)$ for $i \in [N]$ and output $\mathsf{x} = (\mathsf{pk}_i^*, y_i^*)_{i\in[N]}$ and π.

The probability that $\mathcal{P}$ aborts at Step 3 is at most $Q_g(Q_g + Q_s)/2^k$, which is negligible in λ. Unless programming fails, $\mathcal{P}$ perfectly simulates the EU-ACK game for $\mathcal{A}$. The prover $\mathcal{P}$ therefore outputs an accepting proof except with a negligible loss, that is,

$$\mathsf{Adv}_{\mathsf{AS}}^{\mathsf{EU\text{-}ACK}}(\mathcal{A}) \le \Pr\left[\mathcal{V}^{\mathsf{H}}(\mathsf{pp}_{\Pi}, \mathsf{x}, \pi) = 1 \;:\; (\mathsf{x}, \pi) \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in})\right] + \mathsf{negl}(\lambda),$$

where pp_{Π} and $\mathsf{aux\text{-}in}$ are as defined by $\mathcal{B}$. Moreover, if $\mathcal{A}$ breaks EU-ACK there exists some i such that $\mathsf{pk}_i^* = \mathsf{pk}$ and m_i^* was never queried to OSign', implying that $y_i^* = \mathsf{G}'(r_i^*, m_i^*) = \bar{y}_j$ for some $j \in [Q_g]$. Recall $\mathcal{B}$ obtained $\bar{y}_j$ by querying $\mathsf{G}(\bar{r}_j, \bar{m})$ in the EU-CMA$^+$ game, where $\bar{m}$ was never queried to OSign. Our goal is to extract x^* such that $F_{\mathsf{pk}}(x^*) = y_i^*$ and $x^* \in \mathsf{Do}$, which $\mathcal{B}$ can output together with $\bar{r}_j$ and $\bar{m}_j$ to break EU-CMA$^+$. By $\mathcal{Z}$-auxiliary input knowledge soundness, there exists an extractor $\mathcal{E}$ that computes a valid witness for the same distribution of statements as the successful cheating prover $\mathcal{P}$ except with a negligible knowledge error, that is,

$$\begin{aligned}
&\Pr\left[\mathcal{V}^{\mathsf{H}}(\mathsf{pp}_{\Pi}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \mathsf{aux\text{-}in} \leftarrow \mathcal{Z}(1^\lambda), \\ (\mathsf{x}, \pi) \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in}) \end{array}\right] \\
&\le \Pr\left[(\mathsf{x}, \mathsf{w}) \in R_{\mathsf{pp}} \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \mathsf{aux\text{-}in} \leftarrow \mathcal{Z}(1^\lambda), \\ (\mathsf{x}, \pi, \mathsf{aux\text{-}out}, v, \mathsf{w}) \leftarrow \mathcal{E}^{\mathcal{P}^*(\mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in})}(n, \mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in}) \end{array}\right] + \kappa(n, Q_h),
\end{aligned}$$

where in particular w contains x^* such that $F_{\mathsf{pk}}(x^*) = y_i^*$ and $x^* \in \mathsf{Do}$ if the relation is satisfied. Here we rely on $\{(x, \pi, \mathsf{aux\text{-}out}, v) : (x, \pi, \mathsf{aux\text{-}out}) \leftarrow \mathcal{P}^{*\mathsf{H}}(\mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in}) \wedge v \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}_{\Pi}, x, \pi)\}$ and $\{(x, \pi, \mathsf{aux}, v) : (x, \pi, \mathsf{aux}, v, w) \leftarrow \mathcal{E}^{\mathcal{P}^*(\mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in})}(n, \mathsf{pp}_{\Pi}, \mathsf{aux\text{-}in})\}$ being identically distributed, ensuring that we are extracting for the same distribution of statements as the $\mathcal{P}^*$ in EU-ACK proves for. Overall, if $\mathcal{A}$ wins the EU-ACK game, then $\mathcal{B}$ also outputs a signature $(\bar{r}_j, x^*)$ on $\bar{m}$ that verifies w.r.t. pk and G except with a negligible loss. Note that $\mathcal{E}$ is only running in *expected* polynomial time, thus $\mathcal{B}$ only breaks EU-CMA$^+$ security, not EU-CMA security. □

Deterministic Case. For completeness, we also provide an alternative reduction in case the underlying signature scheme is deterministic i.e. $k = 0$ and $\mathsf{S.Sign}^{\mathsf{G}}(\mathsf{sk}, m)$ outputs a unique signature $\sigma = x$ for fixed input (sk, m)[11], which is not covered by the result of [FN16]. Unlike the previous proof, the deterministic case requires a guessing argument of [Cor02] incurring a security loss proportional to the number of signing queries. Our concrete instantiation in later section does not invoke this result, since the standardized Falcon signature does include a sufficiently long random salt. However, the result may be of independent interest if one attempts to construct an aggregate signature scheme from deterministic HtS for the sake of asymptotic compactness.

Lemma C.3. *If the length of salt $k = 0$, S.Sign is deterministic, the message space satisfies $|M| > Q_s + 2Q_g$ and S is EU-CMA$^+$ secure against an adversary that queries Q_g queries to the random oracle G and $Q_s + Q_g$ queries to the signing oracle, and Π is $\mathcal{Z}$-auxiliary input knowledge sound, then AS is EU-ACK secure in the random oracle model against an adversary that makes Q_g queries to the random oracle and Q_s queries to the signing oracle, where $\mathcal{Z}(1^\lambda)$ proceeds as follows: run $\mathsf{pp_S} \leftarrow \mathsf{S.Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{S.Gen}(\mathsf{pp_S})$, $y_i \xleftarrow{\$} \mathsf{Ra}$, $x_i \leftarrow \mathscr{D}(F_{\mathsf{pk}}^{-1}(y_i))$ for $i = 1, \ldots, Q_s + Q_g$, $\bar{y}_i \xleftarrow{\$} \mathsf{Ra}$ for $i = 1, \ldots, Q_g$ and output $(\mathsf{pp_S}, \mathsf{pk}, x_1, y_1, \ldots, x_{Q_s+Q_g}, y_{Q_s+Q_g}, \bar{y}_1, \ldots, \bar{y}_{Q_g})$. The reduction incurs an additional $O(Q_s)$ multiplicative loss in the advantage of breaking EU-CMA$^+$ security.*

Proof. The proof is analogous to Lemma C.2, but we change the way queries to G' and OSign' are answered by $\mathcal{P}$. This is because the sign oracle cannot program the random oracle on the fly anymore due to a lack of random salt. To remedy the situation, we adapt the probabilistic random oracle response routine of [Cor02].

The reduction $\mathcal{B}$ playing the EU-CMA$^+$ game, upon receiving $(\mathsf{pp_S}, \mathsf{pk})$, first obtains $\bar{y}_i$ for $i = 1, \ldots, Q_g$ from G by querying G with distinct messages $\bar{m}_i \in M$. Then $\mathcal{B}$ picks distinct m_i for $i = 1, \ldots, Q_s + Q_g$ (all of which are also different from $\bar{m}_i$) and queries OSign $Q_s + Q_g$ times with m_i to retrieve signatures $(x_i)_{i\in[Q_s+Q_g]}$ such that $\mathsf{G}(m_i) = y_i$, $F_{\mathsf{pk}}(x_i) = y_i$ and $x_i \in \mathsf{Do}$. $\mathcal{B}$ prepares pp_{Π} for the proof system by

[11] In practice, this can be realized by making the Sign algorithm stateful, or by deriving random coins ρ for SampleD deterministically using PRF i.e. $\rho = \mathrm{PRF}_K(m)$ where a secret key K is stored as part of sk.

running $\mathsf{pp}_\Pi \leftarrow \Pi.\mathcal{G}(1^\lambda)$ and defines the auxiliary input $\mathsf{aux\text{-}in} = (\mathsf{pp_S}, \mathsf{pk}, x_1, y_1, \ldots, x_{Q_s+Q_g}, y_{Q_s+Q_g}, \bar{y}_1, \ldots, \bar{y}_{Q_g})$.

The reduction $\mathcal{B}$ then constructs the following cheating prover $\mathcal{P}$ that simulates the view of $\mathcal{A}$ playing the $\mathsf{EU\text{-}ACK}$ game.

1. Initialize $j = k = 0$ and an empty table T.
2. Run $\mathcal{A}$ on input $((\mathsf{pp}_\Pi, \mathsf{pp_S}), \mathsf{pk})$.
3. Upon receiving a query to H' from $\mathcal{A}$, relay that query to H and forward the response from H to $\mathcal{A}$.
4. Upon receiving an ith fresh query $\bar{m}'_i$ to G' from $\mathcal{A}$, flip a biased coin that comes out heads with probability p. If the coin comes out head, increment j, consume (x_j, y_j) in $\mathsf{aux\text{-}in}$ to program the random oracle $\mathsf{G}'(\bar{m}'_i) := y_j$, and record $(\bar{m}'_i, x_j, y_j)$ in the table T; else, increment k and consume $\bar{y}_k$ of $\mathsf{aux\text{-}in}$ to set $\mathsf{G}'(\bar{m}'_i) := \bar{y}_k$
5. Upon receiving an ith query with message m'_i to OSign' from $\mathcal{A}$, if $\mathsf{G}'(m'_i)$ is undefined, increment j, consume (x_j, y_j) in $\mathsf{aux\text{-}in}$ to program the random oracle $\mathsf{G}'(m'_i) := y_j$, record $(\bar{m}'_i, x_j, y_j)$ in the table T, and return x_j as a signature. Else, if $\exists (m'_i, x, y) \in T$ for some (x, y), return x as a signature, and abort otherwise.
6. Upon receiving a forgery $(\mathsf{pk}^*_i, m^*_i)_{i\in[N]}$ and $\sigma_{\mathsf{agg}} = \pi$, let $y^*_i = \mathsf{G}'(m^*_i)$ for $i \in [N]$. Let $i^* \in [N]$ be such that $\mathsf{pk}_{i^*} = \mathsf{pk}$ and $m^*_{i^*}$ was never queried to OSign'. If $\exists (m^*_{i^*}, x, y) \in T$ for some (x, y), abort; else, output $\mathsf{x} = (\mathsf{pk}^*_i, y^*_i)_{i\in[N]}$ and π.

Unless it aborts, $\mathcal{P}$ perfectly simulates the $\mathsf{EU\text{-}ACK}$ game for $\mathcal{A}$. The probability that $\mathcal{P}$ outputs an accepting proof is

$$p^{Q_s} \cdot (1-p) \cdot \mathsf{Adv}^{\mathsf{EU\text{-}ACK}}_{\mathsf{AS}}(\mathcal{A}) \leq \Pr\left[\mathcal{V}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 1 \ : \ (\mathsf{x}, \pi) \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{aux\text{-}in})\right],$$

because $\mathcal{P}$ does *not* abort if the biased coin comes out heads for all Q_s signing queries *and* the coin comes out tail for a query $\mathsf{G}'(m^*_{i^*})$ used for the forgery. By setting $p = Q_s/(Q_s+1)$, since $(1/(1+1/Q_s))^{Q_s} \geq 1/e$ for $Q_s \geq 0$, we have that

$$\frac{1}{e \cdot (Q_s+1)} \cdot \mathsf{Adv}^{\mathsf{EU\text{-}ACK}}_{\mathsf{AS}}(\mathcal{A}) \leq \Pr\left[\mathcal{V}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{x}, \pi) = 1 \ : \ (\mathsf{x}, \pi) \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}_\Pi, \mathsf{aux\text{-}in})\right].$$

The reduction $\mathcal{B}$ then invokes a knowledge extractor $\mathcal{E}$ on $\mathcal{P}$ as in Lemma C.2 and succeeds in outputting a preimage x^* such that $F_{\mathsf{pk}}(x^*) = \bar{y}_k = \mathsf{G}(\bar{m}_k)$ for some $k \in [Q_g]$, where $\bar{m}_k \neq m_j$ for any j thanks to the way $\mathcal{B}$ picked messages. Overall, the probability that $\mathcal{B}$ successfully outputs a forgery is at least

$$\frac{1}{e \cdot (Q_s+1)} \cdot \mathsf{Adv}^{\mathsf{EU\text{-}ACK}}_{\mathsf{AS}}(\mathcal{A}) - \kappa(n, Q_h).$$

□

D Omitted Details of Section 3

D.1 How about fully-splitting?

One may wonder why we have opted for the almost-fully-splitting regime and not for the fully-splitting setting, which provides even faster computation times (for similar-sized moduli). There are multiple arguments supporting our choice, as we detail out in the following. One of our main goals is to keep the LaBRADOR modulus q' at a reasonable size for implementations. Concretely, we do not want to cross the 64-bits barrier. In the fully-splitting regime, a CRT-slot only contains q' many elements, which is then too small to exceed the 2^λ barrier for $\lambda > 64$. Furthermore, the well-spreadness B cannot be smaller than $1/q'$, which is in turn too big to guarantee that challenge elements are non-invertible with probability at most $2^{-\lambda}$ for $\lambda > 64$. To solve this problem, one can either increase the modulus size or use tricks like in [ENS20]. The latter case leads to significant larger proof size as several responses have to be sent. Moreover, there might be subtleties that arise when showing adaptive knowledge soundness of non-interactive LaBRADOR in combination with the advanced tricks. The first case, i.e., increasing the modulus to a λ-bit size, leads to actual computation performance losses, as we show in Section 6.2.

D.2 Proof of Lemma 3.1

Proof. Recall that a ring element is invertible, if and only if none of its CRT-slots are zero. Thus

$$\begin{aligned}
\Pr_{\boldsymbol{c} \xleftarrow{\$} \mathcal{C}} [\boldsymbol{c} - \boldsymbol{y} \in \mathcal{R}_q^\times] &= \Pr[\boldsymbol{c} - \boldsymbol{y} \bmod (X^\delta - \zeta_i) \neq \boldsymbol{0} \ \forall i \in [l]] \\
&= 1 - \Pr[\exists i \in [l] \colon \boldsymbol{c} - \boldsymbol{y} \bmod (X^\delta - \zeta_i) = \boldsymbol{0}] \\
&\geq 1 - \sum_{i=1}^{l} \Pr[\boldsymbol{c} - \boldsymbol{y} \bmod (X^\delta - \zeta_i) = \boldsymbol{0}] \\
&= 1 - \sum_{i=1}^{l} \Pr[\boldsymbol{c} \bmod (X^\delta - \zeta_i) = \boldsymbol{y}] \\
&\geq 1 - l \cdot B,
\end{aligned}$$

where we applied the union bound in the third to last inequality and the property of B-well-spreadness in the last inequality. ◻

D.3 Proof of Lemma 3.2

Proof. The proof is by induction over n. We start with the base case, where $n = 1$. Since $\boldsymbol{f}$ is non-zero, there exists $j \in [l]$, such that $\boldsymbol{f} \bmod (X^\delta - \zeta_j)$ is non-zero. Further, for any $\boldsymbol{c} \in \mathcal{R}_q$ it yields

$$\begin{aligned}
\Pr[\boldsymbol{f}(\boldsymbol{c}) = \boldsymbol{0}] &= \Pr[\boldsymbol{f}(\boldsymbol{c}) \bmod (X^\delta - \zeta_i) = \boldsymbol{0} \ \forall i \in [l]] \\
&\leq \Pr[\boldsymbol{f}(\boldsymbol{c}) \bmod (X^\delta - \zeta_j) = \boldsymbol{0}].
\end{aligned}$$

As $\boldsymbol{f} \bmod (X^\delta - \zeta_j)$ is a polynomial over the field $\mathbb{Z}_q[X]/(X^\delta - \zeta_j)$, it has at most $D := \deg(\boldsymbol{f} \bmod (X^\delta - \zeta_j))$ distinct roots. Moreover, the degree after reducing modulo $(X^\delta - \zeta_j)$ cannot increase, hence $D \leq \deg(\boldsymbol{f})$. The probability that $\boldsymbol{f} \bmod (X^\delta - \zeta_j)$ is zero cannot decrease if we assume that all of its roots are in $\mathcal{C} \bmod X^\delta - \zeta_j$. Moreover, by the well-spreadness of $\mathcal{C}$, the probability that the j-th CRT-slot of a random challenge element $\boldsymbol{c}$ hits one of those roots is at most B. Thus, by a union bound over the D roots of $\boldsymbol{f} \bmod (X^\delta - \zeta_j)$, it yields

$$\Pr[\boldsymbol{f}(\boldsymbol{c}) = \boldsymbol{0}] \leq D \cdot B \leq \deg(\boldsymbol{f}) \cdot B.$$

Now, assume that the lemma holds for all polynomials in $n - 1$ variables and let $\boldsymbol{f}(X_1, \ldots, X_n)$ be a polynomial in n variables. We can interpret $\boldsymbol{f}$ as a polynomial in X_1 by writing it as

$$\boldsymbol{f}(X_1, \ldots, X_n) = \sum_{i=0}^{\deg(\boldsymbol{f})} \boldsymbol{f}_i(X_2, \ldots, X_n) \cdot X_1^i,$$

where all the $\boldsymbol{f}_i$ are polynomials in $n-1$ variables. As $\boldsymbol{f}$ is non-zero, there exists an index $j \in \{0, \ldots, \deg(\boldsymbol{f})\}$ such that $\boldsymbol{f}_i$ is non-zero. Let j be the largest of such indices. Since the degree of $\boldsymbol{f}_j \cdot X_1^j$ is at most $\deg(\boldsymbol{f})$, we know that $\boldsymbol{f}_j$ has degree at most $\deg(\boldsymbol{f}) - j$. We now sample $\boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$ independently and uniformly at random from $\mathcal{C}$. By the induction hypothesis, $\Pr[\boldsymbol{f}_j(\boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) = \boldsymbol{0}] \leq (\deg(\boldsymbol{f}) - j) \cdot B$. Assuming that $\boldsymbol{f}_j(\boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) \neq \boldsymbol{0}$, then $\boldsymbol{f}(X_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n)$ is a polynomial in one variable and of degree j. By the induction base case, for $\boldsymbol{c}_1$ sampled uniformly at random from $\mathcal{C}$, it yields $\Pr[\boldsymbol{f}(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) = \boldsymbol{0} \mid \boldsymbol{f}_j(\boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) \neq \boldsymbol{0}] \leq j \cdot B$. We denote the event $\boldsymbol{f}(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) = \boldsymbol{0}$ by E and the event $\boldsymbol{f}_j(\boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) = \boldsymbol{0}$ by F. We observe that

$$\begin{aligned}
\Pr[E] &= \Pr[E \mid F] \cdot \Pr[F] + \Pr[E \mid \neg F] \cdot \Pr[\neg F] \\
&\leq \Pr[F] + \Pr[E \mid \neg F] \\
&\leq (\deg(\boldsymbol{f}) - j) \cdot B + j \cdot B = \deg(\boldsymbol{f}) \cdot B,
\end{aligned}$$

concluding the induction. ◻

D.4 Almost-Fully-Splitting-Instantiation

In the following, we present our choice of $\mathcal{R}_q$ and $\mathcal{C}$, which follows the approach from [ESZ22, ESLR23]. Again, let $\mathcal{R}_q = \prod_{i=1}^{l} \mathbb{Z}_q[X]/\langle X^\delta - \zeta_i \rangle$, where $d = l \cdot \delta$.

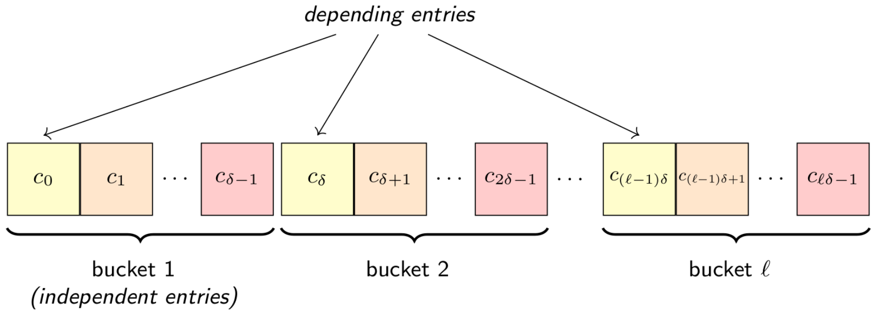


Fig. 5: Let $(c_0, \ldots, c_{l\delta-1})$ be the vector representing a challenge element $\boldsymbol{c} \in \mathcal{C}$ drawn with the partition-and-sample technique [ESZ22], where $d = l \cdot \delta$.

Recall that a challenge element $\boldsymbol{c} \in \mathcal{C} \subset \mathcal{R}_q$ corresponds to a polynomial of degree $d-1$, thus it can be represented by a vector of dimension d over $\mathbb{Z}_q$. The key idea is to group those d coefficients into l buckets of size δ and to sample for $k \in \{0, \ldots, \delta-1\}$ the k-th entries of each bucket depending on each other, while keeping the δ coefficients *within* one bucket independent of each other. The latter property is relevant to prove well-spreadness. In [ESZ22], this is called the *partition-and-sample* technique. We illustrate the idea in Figure 5.

More formally, we set $S_\gamma := \{\tilde{\boldsymbol{c}}(x) = c_0 + c_\delta x^\delta + \cdots + c_{(l-1)\delta} x^{(l-1)\delta} \colon \|\tilde{\boldsymbol{c}}\|_\infty = \gamma\}$ and define $\mathcal{C} := \{\tilde{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_1 x + \cdots + \tilde{\boldsymbol{c}}_{\delta-1} x^{\delta-1} \colon \tilde{\boldsymbol{c}}_k \in S_\gamma \wedge \|\tilde{\boldsymbol{c}}_k\|_1 = \tilde{w} \text{ for } k \in \{0, \ldots, \delta-1\}\}$. As we are enforcing a fixed Hamming weight $\tilde{w}$ on the set of the k-th entries of all buckets, we make them depending on each other. Overall, the Hamming weight of a challenge element $\boldsymbol{c} \in \mathcal{C}$ is given by $w = \delta \cdot \tilde{w}$. Modifying $\tilde{w}$, thus modifying the level of dependency, helps us to control the size and well-spreadness of the challenge space $\mathcal{C}$ and the different norms of the challenge elements. Note that [ESZ22] originally proposed this challenge space for $\gamma = 1$ and [ESLR23] generalized it to arbitrary $\gamma \geq 1$.

Well-spreadness. Implicitly, [ESLR23] already proved B-well-spreadness of their challenge space, as summarized in the lemma below.

Lemma D.1 (Adapted from [ESLR23, Lem. 1]). *Given the parameters d, q, l, δ, γ and $\tilde{w}$, the (heuristically) expected well-spreadness of $\mathcal{C}$ is given by $\mathbb{E}[B] = \mathbb{E}[M_2]^\delta$, where $\mathbb{E}[M_2] = \eta \left(\frac{1}{q} + \left(1 - \frac{1}{q}\right) A(\tilde{w}, \gamma l) \right)$ for $\eta := \frac{l^{\tilde{w}} (l-\tilde{w})!}{l!}$ and $A(\tilde{w}, \gamma l) = \Gamma\left(\frac{\tilde{w}+1}{2}\right) \frac{1}{\sqrt{\pi \cdot (l\gamma)^{\tilde{w}}}}$.*

As in [ESZ22], we simplified the concrete formula for the well-spreadness by heuristically modelling powers of primitive roots of unities as independent uniformly random elements in $\mathbb{Z}_q$. The function $A(\tilde{w}, \gamma l)$ is the $\tilde{w}$'th *central absolute moment* of a normally distributed variable with standard deviaiton $1/\sqrt{2\gamma l}$, cf. [Win14, Equation 18] and $\Gamma(\cdot)$ the gamma function. For smaller sized moduli (up to 35-bits), we were able to check the heuristic results with the exact results, but our computation capacities could not handle the large moduli we need for our parameter setting.

We refer to their paper for the precise formula of B. On a high level, the partition-and-sample technique makes sure that the depending coefficients of $\boldsymbol{c}$ will all contribute to the same coefficient within a given CRT-slot. Thus, the coefficients of every CRT-slot of $\boldsymbol{c}$, which is given by a polynomial of degree δ, are independent of each other. The strategy of [ESLR23] is to provide a concrete bound M_2 for the probability that *one* of the δ coefficients in a given CRT-slot hits a specific field element in $\mathbb{Z}_q$. As all δ coefficients are independent from each other by the design of $\mathcal{C}$, the probability that a CRT-slot hits a specific element in $\mathbb{Z}_q^\delta$ is bounded above by $B = M_2^\delta$. In our case, the parameters d, q, δ and l are fixed, so we evaluate the function from Lemma D.1 to compute the smallest infinity norm bound γ and weight $\tilde{w}$ such that $l \cdot B \leq 2^{-\lambda}$ for a given security parameter λ.

Concretely, for our instantiation of LaBRADOR for Falcon-512 signatures with security level $\lambda = 128$, ring degree $d = 512$, Falcon modulus $q = 12289$, LaBRADOR modulus $q' \in [2^{47}, 2^{63}]$, split ratio $\delta = 4$ and split factor $l = 512/4$, we obtain a minimal infinity norm bound $\gamma = 1$ and weight $\tilde{w} = 14$, leading to an overall weight of $w = \delta \cdot \tilde{w} = 56$. For Falcon-1024 signatures with $\lambda = 256$, $d = 1024$, $q = 12289$, $q' \in [2^{47}, 2^{63}]$, $\delta = 8$ and $l = 1024/8$, we obtain a minimal infinity norm bound $\gamma = 1$, weight $\tilde{w} = 13$, leading to an overall weight of $w = \delta \cdot \tilde{w} = 104$. When applying the subring trick (cf. Section 5.4), one can go down to $d' = 128$ for $\lambda = 128$ and to $d' = 256$ for $\lambda = 256$, leading to slightly larger weights and norm bounds.

We also provide weights and infinity norm bounds for the two-splitting regime. Here, one can set $B = 1/|\mathcal{C}|$ (as this is the probability that the difference of two challenge elements is invertible). We summarize all parameters in Table 3.

Size of challenge space. For fixed δ and l, the choice of $\tilde{w}$ and γ determines the size of $\mathcal{C}$, as it yields $|\mathcal{C}| = (\binom{l}{\tilde{w}}(2\gamma)^{\tilde{w}})^{\delta}$. In the knowledge soundness proof of LaBRADOR we need $|\mathcal{C}| \geq 2^{\lambda} \cdot \mu$, where λ is the security parameter and $\mu = (5 + 2l)r$ is defined by the split factor l and an upper bound on the number of witnesses r in each round. It turns out that this requirement is less restrictive than the condition to guarantee a small enough well-spreadness B from above.

Size of CRT-slot. The size of each of the l CRT-slots is given by q^{δ}. In the knowledge soundness proof of LaBRADOR we require $q^{\delta} \geq 2^{\lambda}$ for a aimed security level λ. As long as q is at least 32-bits long, this requirement is fulfilled for both Falcon-512 and Falcon-1024.

Norm bounds on challenges. By construction, for every $\boldsymbol{c} \in \mathcal{C}$, it yields $\|\boldsymbol{c}\|_1 \leq w \cdot \gamma$, $\|\boldsymbol{c}\|_\infty \leq \gamma$ and $\|\boldsymbol{c}\|_2 \leq \sqrt{w}\gamma$. Thus, $T_2 \leq w\gamma^2$. We are also interested in a bound T_{op} on the operator norm. Note that in power-of-two cyclotomic rings it holds $\|\boldsymbol{r} \cdot \boldsymbol{s}\|_2 \leq \|\boldsymbol{r}\|_1 \cdot \|\boldsymbol{s}\|_2$ for every two ring elements $\boldsymbol{r}, \boldsymbol{s} \in \mathcal{R}$. Thus, $\|\boldsymbol{cs}\|_2 / \|\boldsymbol{s}\|_2 \leq \|\boldsymbol{c}\|_1 \leq w \cdot \gamma$, implying that $\|\boldsymbol{c}\|_{\text{op}} \leq w \cdot \gamma$, thus we can set $T_{\text{op}} \leq w \cdot \gamma$. We use the same approach as in LaBRADOR and assume slightly tighter bounds by rejection sampling on challenge elements. More precisely, we set $T_2 := w\gamma/c$ and $T_{\text{op}} := w\gamma^2/c$ for $c = 2$ if $\lambda = 128$ and $c = 2.5$ if $\lambda = 256$.

Splitting	d'	l	λ	$\log_2 q'$	w	γ
Almost-fully	128	32	128	$\in [47, 63]$	64	4
	256	32	256	$\in [47, 63]$	144	3
Two	64	32	128	*arbitrary*	43	2
	128	64	256	*arbitrary*	74	2

Table 3: Resulting weight w and infinity norm bound γ for the two different regimes for the challenge set $\mathcal{C}$, two-splitting and almost-fully-splitting. By d' and l we denote the degree and split factor of the chosen quotient subring $\mathcal{S}$ modulo q'. The parameters are set such that the challenge size is larger than $2^{\lambda}\mu$ and the well-spreadness bound B is smaller than $2^{-\lambda}/\mu$, where λ is the targeted security level and $\mu = (5 + 2l)r$ is defined by the split factor l and an upper bound on the number of witnesses r in each round.

E Detailed Performance Comparison

Aimed Security Level. All size estimates of this section are targeting a concrete security level λ_{final} by assuming that at most $N \leq 10\,000$ signatures are aggregated, as we believe this is a realistic scenario. By Lemma C.2, assuming λ_{sig} bits of security for the underlying signature scheme and λ_{snark} bits of knowledge error for the used SNARK, we define $\lambda_{\mathsf{final}} = \min\{\lambda_{\mathsf{sig}}, \lambda_{\mathsf{snark}}\}$ bits of security for our aggregate signature scheme. Recall that the signature schemes Falcon-512 and Falcon-1024 provide $\lambda_{\mathsf{sig}} = 128$ and $\lambda_{\mathsf{sig}} = 256$ bits of security, respectively. Equations 2+3 in Theorem 4.2 and Equation 18 in Section I.5 define λ_{snark}. Note that we ignore the security loss caused by the extractor's runtime in Theorem 4.2 and thus set $Q = 0$. This is a common practice when setting concrete parameters and can be compared to ignoring the security loss caused by the forking lemma, as for instance done in the aggregate signature [FSZ22]. Moreover, for $N \leq 10\,000$, we observe at most $t = 8$ recursions of LaBRADOR. Equations 2+3 are composed of 6 additive terms. Our strategy is to upper bound every single term by $2^{-\lambda}$ for some λ, leading to $2^{-\lambda_{\mathsf{snark}}} \leq 2 \cdot t \cdot 6 \cdot 2^{-\lambda} \leq 2^{-\lambda+7}$. Concretely, we set $\lambda = 128$ for Falcon-512 aggregation and $\lambda = 256$ for Falcon-1024 aggregation. Hence, overall, our aggregate signature scheme for Falcon-512 signatures guarantees a security level of 121 bits, whereas the aggregate signature scheme for Falcon-1024 signatures guarantees 249 bits of security.

The original LaBRADOR protocol aimed at $\lambda = 128$. Increasing λ to 256 (or other values) has an impact on all additive terms of Equations 2+3. First and foremost, the two $\mathsf{M\text{-}SIS}$ problems need to be parameterized such that their concrete security is λ. Moreover, the challenge space $\mathcal{C}$ needs to be of size at least 2^{λ} and its well-spreadness bound B should lead to $(5 + 2l)Br \leq 2^{-\lambda}$, where l is the split factor

Aggr. Signature Scheme	# Sign. N	Sec. Param. λ	$\lvert\sigma_{\text{agg}}\rvert$
[JRS23, Tab. 4]	500	128	3616 kB
Ours for Falcon-512 with salt	500	121	**138** kB
Ours for Falcon-512 without salt	500	121	118 kB
[JRS23, Tab. 4]	1000	128	7444 kB
Ours for Falcon-512 with salt	1000	121	**164** kB
Ours for Falcon-512 without salt	1000	121	124 kB
[JRS23, Tab. 4]	2000	128	15319 kB
Ours for Falcon-512 with salt	2000	121	**215** kB
Ours for Falcon-512 without salt	2000	121	135 kB

Table 4: Comparison of our Falcon-512 aggregate signature sizes (with and without salt) with aggregate signature [JRS23] for different parameters.

of the subring $\mathcal{S}_{q'}$ and $r = 6\lceil\sqrt{N}\rceil + 1$, an upper bound on the number of witnesses in each iteration. Further, the security level defines the dimensions and constants in the Johnson-Lindenstrauss projections (Lemma 2.2) and defines the dimensions of the aggregation vector $\vec{\beta}$ in Step 3 of LaBRADOR. If future use cases of LaBRADOR require a different security level than $\lambda \in \{128, 256\}$, all those factors need to be taken into account.

E.1 Comparison With [JRS23]

To the best of our knowledge, there is only one lattice-based non-interactive aggregate signature scheme, proven in the same non-interactive security model as ours, providing non-trivial compression as well as concrete numbers [JRS23]. We emphasize that their aggregation only works for GPV-style signatures that use the MP-trapdoors [MP12] with LW-sampler [LW15]. Hence, their construction does not work for Falcon. They provide benchmarks [JRS23, Table 4] for the number of signatures N going up to 500, 1000 and 2000 and target a security level of 128, whereas we target 121 bits of security in our aggregation of Falcon-512. In Table 4, one can see that our aggregate signatures are of multiple orders of magnitude smaller than the ones from [JRS23]. Moreover, their aggregation does not lead to smaller aggregate signature sizes for large numbers of signatures, in particular beyond $N \geq 4000$. Note that their starting signature is also significantly larger than a Falcon-512 signature, to begin with.

E.2 Comparison With Squirrel and Chipmunk

Squirrel [FSZ22] and its recent optimization Chipmunk [FHSZ23] are multi-signature schemes in the synchronized setting whose security also relies on M-SIS. Both can be seen as an aggregate signature scheme which aggregates signatures for possibly distinct parties and messages but which were issued for the *same time step*. Due to their tree-based construction, their multi signature schemes can only be used for a certain amount of time (which [FSZ22] calls the *life cycle* of their scheme). All this are significant restrictions which our aggregate signature does not incur. Due to the lack of other concrete results on aggregate signatures, we still think it is instructive to compare the estimates of our aggregate signature sizes with the ones from Squirrel and Chipmunk. Squirrel provides benchmarks [FSZ22, Table 3] for the number of signatures N being 1024, 4096 and 8192 and life cycles of 8 months, 5 years and 12 years. Chipmunk provides benchmarks [FHSZ23, Table 5] for the number of signatures N being 1024 and 8192 and the same life cycles. Both target a security level of 112, whereas we target a security level of 121 for Falcon-512. Table 5 shows that our aggregate scheme produces smaller aggregate signatures than Squirrel but larger ones than Chipmunk for every set of parameters. However, for $N = 1024$, the difference between Chipmunk and our aggregate signature is quite small. Further note that Chipmunk signatures are significantly larger than a Falcon-512 signature, to begin with.

F Full Description of Padded Falcon Aggregation Constraints

F.1 Reformulating Constraints for Better Recursion

Padding Scheme. We first present the padding scheme generically. Say that we have the elements $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N \in \mathcal{R}_{q'}$ in the old witness, along with the elements $\boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_N \in \mathcal{R}_{q'}$ such that $\|\boldsymbol{w}_i\|_2^2 = \mathsf{ct}\left(\boldsymbol{w}'_i \boldsymbol{w}_i\right)$.

Aggr. Signature Scheme	# Sign. N	Sec. Param. λ	Life Cycle	$\lvert\sigma_{\mathsf{agg}}\rvert$
Squirrel [FSZ22]	1024	112	8 months	572 kB
	1024	112	5 years	635 kB
	1024	112	21 years	677 kB
Chipmunk [FHSZ23]	1024	112	**8 months**	**118** kB
	1024	112	5 years	133 kB
	1024	112	21 years	143 kB
Ours for Falcon-512 with salt	1024	121	∞	**165** kB
Ours for Falcon-512 without salt	1024	121	∞	124 kB
Squirrel [FSZ22]	4096	112	8 months	693 kB
	4096	112	5 years	711 kB
	4096	112	21 years	823 kB
Chipmunk [FHSZ23]	/	/	/	/
Ours for Falcon-512 with salt	4096	121	∞	**290** kB
Ours for Falcon-512 without salt	4096	121	∞	126 kB
Squirrel [FSZ22]	8192	112	8 months	762 kB
	8192	112	5 years	820 kB
	8192	112	21 years	908 kB
Chipmunk [FHSZ23]	8192	112	**8 months**	**160** kB
	8192	112	5 years	180 kB
	8192	112	21 years	194 kB
Ours for Falcon-512 with salt	8192	121	∞	**458** kB
Ours for Falcon-512 without salt	8192	121	∞	130 kB

Table 5: Comparison of our Falcon-512 aggregate signature sizes (with and without salt) with synchronized multi-signature Squirrel [FSZ22] and its optimization Chipmunk [FHSZ23] for different parameters. The ∞ symbol indicates that there is no a-priori bound on the life cycle of our aggregate signature scheme.

Define $\rho = \lfloor \sqrt{N} \rceil$. In the new witness, we are going to have $\lceil \frac{N}{\rho} \rceil$ vectors $\vec{\boldsymbol{u}}_1, \ldots, \vec{\boldsymbol{u}}_{\lceil \frac{N}{\rho} \rceil} \in \mathcal{R}_{q'}^N$ containing the $\boldsymbol{w}_i$, and ρ vectors $\vec{\boldsymbol{u}}'_1, \ldots, \vec{\boldsymbol{u}}'_\rho \in \mathcal{R}_{q'}^N$ containing the $\boldsymbol{w}'_i$. We define the vectors such that

$$(\vec{\boldsymbol{u}}_i)_j = \begin{cases} \boldsymbol{w}_i & \text{if } (i-1)\rho < j \leq i\rho \\ 0 & \text{else} \end{cases} \quad \text{and} \quad (\vec{\boldsymbol{u}}_i)_j = \begin{cases} \boldsymbol{w}'_j & \text{if } j \equiv i \bmod \rho \\ 0 & \text{else} \end{cases}.$$

The padding scheme is best described visually, see Figure 6. To keep track of where the $\boldsymbol{w}_i$, $\boldsymbol{w}'_i$ are stored, we define the index functions

$$\text{index}(i) = \left\lceil \frac{i}{\rho} \right\rceil \quad \text{and} \quad \text{index}'(i) = (i - 1 \bmod \rho) + 1.$$

Then $\boldsymbol{w}_i$ is stored at $(\vec{\boldsymbol{u}}_{\text{index}(i)})_i$ and $\boldsymbol{w}'_i$ is stored at $(\vec{\boldsymbol{u}}'_{\text{index}'(i)})_i$. Importantly, except for $\vec{\boldsymbol{u}}_{\text{index}(i)}$ and $\vec{\boldsymbol{u}}'_{\text{index}'(i)}$, the i-th entry of each vector is always $\mathbf{0}$. From this, it follows that

$$\mathsf{ct}\left(\langle \vec{\boldsymbol{u}}_{\text{index}(i)}, \vec{\boldsymbol{u}}'_{\text{index}'(i)} \rangle\right) = \mathsf{ct}\left(\boldsymbol{w}_i \boldsymbol{w}'_i\right) = \|\boldsymbol{w}_i\|_2^2.$$

Final Witness and Constraints. After applying the padding scheme transformation, we end up with $r = 3\lceil \frac{N}{\rho} \rceil + 3\rho + 1$ witness vectors of rank $n = N$:

- $\vec{\boldsymbol{y}}_{1,j}, \ldots, \vec{\boldsymbol{y}}_{\lceil \frac{N}{\rho} \rceil, j}$ and $\vec{\boldsymbol{y}}'_{1,j}, \ldots, \vec{\boldsymbol{y}}'_{\rho,j}$ for $j = 1, 2$: The padding of $\boldsymbol{s}_{1,j}, \ldots, \boldsymbol{s}_{N,j}$ and $\boldsymbol{s}'_{1,j}, \ldots, \boldsymbol{s}'_{N,j}$.
- $\vec{\boldsymbol{e}}_1, \ldots, \vec{\boldsymbol{e}}_{\lceil \frac{N}{\rho} \rceil}$ and $\vec{\boldsymbol{e}}'_1, \ldots, \vec{\boldsymbol{e}}'_\rho$: The padding of $\boldsymbol{\varepsilon}_1, \ldots, \boldsymbol{\varepsilon}_N$ and $\boldsymbol{\varepsilon}'_1, \ldots, \boldsymbol{\varepsilon}'_N$.
- $\vec{\boldsymbol{v}}$: A single vector collecting the $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$.

The old constraints need to be transformed to fit the new witness. Let $\vec{\boldsymbol{\delta}}_i \in \mathcal{R}_{q'}^N$ be the vector that has i-th entry $\mathbf{1}$ and all other entries $\mathbf{0}$. Then we can formulate the i-th Falcon verification equation as

$$\langle \vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{y}}_{\text{index}(i),1} \rangle + \langle \boldsymbol{h}_i \vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{y}}_{\text{index}(i),2} \rangle + \langle q \vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{v}} \rangle - \boldsymbol{t}_i = \mathbf{0} \bmod q'.$$

The new i-th four-square constraint is

$$\mathsf{ct}\left(\langle \vec{\boldsymbol{y}}_{\text{index}(i),1}, \vec{\boldsymbol{y}}'_{\text{index}'(i),1} \rangle + \langle \vec{\boldsymbol{y}}_{\text{index}(i),2}, \vec{\boldsymbol{y}}'_{\text{index}'(i),2} \rangle + \langle \vec{\boldsymbol{e}}_{\text{index}(i)}, \vec{\boldsymbol{e}}'_{\text{index}'(i)} \rangle - \beta^2\right) = 0 \bmod q'.$$

The dot product constraints for the $\boldsymbol{s}'_{i,1}, \boldsymbol{s}'_{i,2}$, $\boldsymbol{\varepsilon}_i$, $\boldsymbol{\varepsilon}'_i$ are all of the form (7), checking that a single $\mathbb{Z}_{q'}$-coefficient of a witness element is equal to a constant or that it is equal to another coefficient in the witness. To check that j-th $\mathbb{Z}_{q'}$-coefficient of the i-th entry of some witness vector $\vec{\boldsymbol{a}} \in \mathcal{R}_{q'}^N$ is equal to some constant $b \in \mathbb{Z}_{q'}$, or to check that it is equal to the j'-th coefficient of the i'-th entry of some other witness vector $\vec{\boldsymbol{c}} \in \mathcal{R}_{q'}^N$, we add a constraint of the form

$$\mathsf{ct}\left(\langle \sigma_{-1}(X^j)\vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{a}}\rangle - b\right) = 0 \bmod q' \quad \text{or} \quad \mathsf{ct}\left(\langle \sigma_{-1}(X^j)\vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{a}}\rangle - \langle \sigma_{-1}(X^{j'})\vec{\boldsymbol{\delta}}_{i'}, \vec{\boldsymbol{c}}\rangle\right) = 0 \bmod q'.$$

We also need to add constraints to enforce that the other entries of the witness are 0 in accordance with the padding scheme. To check that the i-th entry of some witness vector $\vec{\boldsymbol{a}} \in \mathcal{R}_{q'}^N$ is $\mathbf{0}$, we add the constraint

$$\langle \vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{a}}\rangle = \mathbf{0} \bmod q'.$$

Finally, for the projections in the first iteration, nothing really changes. We only need to project the entries that are not 0 in the padding scheme, because we check that the other entries are indeed 0 and contribute nothing to the norm of the witness vectors. Hence, $\vec{p}_1$ and $\vec{p}_2$ are random linear projections of the same elements as before, meaning that we can keep the the bounds from the previous section.

Impact on Runtime and Proof Size. To see the benefits of the new formulation, let us first analyze the formulation we had at end of Section 5.2. With $r = O(N)$ witness elements, there are $O(r^2) = O(N^2)$ garbage polynomials in the first iteration, which the prover must compute. With our set of dot product constraints, the runtime of the prover in the k-th iteration is $O\left(n_k r_k + m_k\right)$. With $m_1 = O(N)^2$, we get a $O\left(N^2\right)$ runtime for the prover. Furthermore, since $n_1 \ll m_1$, the recursion does not start from a balanced state. Recursing after the first iteration, the new rank is $n_2 = \max(\frac{1}{\nu}, \frac{m_1}{\mu}) = \frac{m_1}{\mu}$, where ν and μ are the folding parameters when going from one iteration level to the next. Effectively, this means that recursion with the old formulation starts at rank $O(N^2)$.

F.2 Final constraints before moving to subring

Define $\vec{\boldsymbol{\delta}}_i \in \mathcal{R}_{q'}^n$ to be the vector with $(\vec{\boldsymbol{\delta}}_i)_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$.

Main constraints:

For $i = 1$ to n:

$$\langle \vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{y}}_{\text{index}(i),1}\rangle + \langle \boldsymbol{h}_i\vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{y}}_{\text{index}(i),2}\rangle + \langle q\vec{\boldsymbol{\delta}}_i, \vec{\boldsymbol{v}}\rangle - \mathsf{H}(m_i) = \mathbf{0} \bmod q' \tag{Falcon eq}$$

$$\mathrm{ct}\left(\left(\sum_{j=1}^{2}\langle \vec{\boldsymbol{y}}_{\text{index}(i),j}, \vec{\boldsymbol{y}}'_{\text{index}'(i),j}\rangle\right) + \langle \vec{\boldsymbol{e}}_{\text{index}(i)}, \vec{\boldsymbol{e}}'_{\text{index}'(i)}\rangle - \beta^2\right) = 0 \bmod q' \tag{4 squares}$$

$$\|(\vec{\boldsymbol{y}}_{1,1}, \ldots, \vec{\boldsymbol{y}}_{\lceil \frac{n}{\rho}\rceil,2}, \vec{\boldsymbol{y}}'_{1,1}, \ldots, \vec{\boldsymbol{y}}'_{\lceil \frac{n}{\rho}\rceil,2}, \vec{\boldsymbol{e}}_1, \ldots, \vec{\boldsymbol{e}}_\rho, \vec{\boldsymbol{e}}'_1, \ldots, \vec{\boldsymbol{e}}'_\rho)\|_\infty \le \sqrt{\frac{q'}{2(2d+4)}}$$

$$\|\vec{\boldsymbol{v}}\|_\infty \le \frac{q'}{6q}$$

Form constraints

Form constraints for the $\vec{\boldsymbol{y}}_{i,j}$: We check that there has been padded with 0s correctly.
For $i = 1$ to $\lceil \frac{n}{\rho}\rceil$:
For $j = 1$ to 2:
For $k = 1$ to n, $k \notin [(i-1)\rho + 1, i\rho]$:

$$\langle \vec{\boldsymbol{\delta}}_k, \vec{\boldsymbol{y}}_{i,j}\rangle = \mathbf{0} \bmod q'$$

Form constraints for the $\vec{\boldsymbol{y}}'_{i,j}$:
For $i = 1$ to ρ:
For $j = 1$ to 2:
For $k = 1$ to n:
If $k \equiv i \bmod \rho$: We check that $\sigma_{-1}((\vec{\boldsymbol{y}}_{i,j})_k) = (\vec{\boldsymbol{y}}'_{i,j})_k$.

$$\mathrm{ct}\left(\langle \vec{\boldsymbol{\delta}}_k, \vec{\boldsymbol{y}}_{i,j}\rangle + \langle -\vec{\boldsymbol{\delta}}_k, \vec{\boldsymbol{y}}'_{i,j}\rangle\right) = 0 \bmod q'$$

Fig. 6: Illustration of the padding scheme when ρ divides N. The columns of the left matrix are $\vec{\boldsymbol{u}}_1, \ldots, \vec{\boldsymbol{u}}_{\lceil \frac{N}{\rho} \rceil} \in \mathcal{R}_{q'}^N$, and the columns of the right matrix are $\vec{\boldsymbol{u}}'_1, \ldots, \vec{\boldsymbol{u}}'_\rho \in \mathcal{R}_{q'}^N$. When ρ does not divide N, the last couple of entries in the pattern is $\mathbf{0}$.

$$
\left[\vec{\boldsymbol{u}}_1, \ldots, \vec{\boldsymbol{u}}_{\lceil \frac{N}{\rho} \rceil}\right] = \begin{bmatrix}
\boldsymbol{w}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\boldsymbol{w}_\rho & \mathbf{0} & \vdots & \cdots & \vdots \\
\mathbf{0} & \boldsymbol{w}_{\rho+1} & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \boldsymbol{w}_{2\rho} & \mathbf{0} & \cdots & \vdots \\
\vdots & \mathbf{0} & \boldsymbol{w}_{2\rho+1} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \boldsymbol{w}_{3\rho} & \cdots & \vdots \\
\vdots & \vdots & \mathbf{0} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \cdots & \boldsymbol{w}_{N-\rho+1} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{w}_N
\end{bmatrix}, \quad
\left[\vec{\boldsymbol{u}}'_1, \ldots, \vec{\boldsymbol{u}}'_\rho\right] = \begin{bmatrix}
\boldsymbol{w}'_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \boldsymbol{w}'_2 & \mathbf{0} & \cdots & \vdots \\
\vdots & \mathbf{0} & \boldsymbol{w}'_3 & \cdots & \vdots \\
\vdots & \vdots & \mathbf{0} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \mathbf{0} \\
\mathbf{0} & \vdots & \vdots & \cdots & \boldsymbol{w}'_\rho \\
\boldsymbol{w}'_{\rho+1} & \mathbf{0} & \vdots & \cdots & \mathbf{0} \\
\mathbf{0} & \boldsymbol{w}'_{\rho+2} & \mathbf{0} & \cdots & \vdots \\
\vdots & \mathbf{0} & \boldsymbol{w}'_{\rho+3} & \cdots & \vdots \\
\vdots & \vdots & \mathbf{0} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \mathbf{0} \\
\mathbf{0} & \vdots & \vdots & \cdots & \boldsymbol{w}'_{2\rho} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\boldsymbol{w}'_{N-\rho+1} & \mathbf{0} & \vdots & \cdots & \mathbf{0} \\
\mathbf{0} & \boldsymbol{w}'_{N-\rho+2} & \mathbf{0} & \cdots & \vdots \\
\vdots & \mathbf{0} & \boldsymbol{w}'_{N-\rho+3} & \cdots & \vdots \\
\vdots & \vdots & \mathbf{0} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{w}'_N
\end{bmatrix}
$$

For $l = 1$ to $d - 1$:

$$\mathrm{ct}\Big(\langle \sigma_{-1}(X^l)\vec{\boldsymbol{\delta}}_k, \vec{\boldsymbol{y}}_{i,j}\rangle + \langle \sigma_{-1}(X^{d-l})\vec{\boldsymbol{\delta}}_k, \vec{\boldsymbol{y}}'_{i,j}\rangle\Big) = 0 \bmod q'$$

Else:

$$\langle \vec{\boldsymbol{\delta}}_k, \vec{\boldsymbol{y}}'_{i,j}\rangle = \mathbf{0} \bmod q'$$

<u>Form constraints for the $\vec{\boldsymbol{e}}_i$:</u>
For $i = 1$ to $\lceil \frac{n}{\rho} \rceil$:
For $j = 1$ to n:
If $(i-1)\rho < j \leq i\rho$:
For $k = 4$ to $d - 1$:

$$\mathrm{ct}\Big(\langle \sigma_{-1}(X^k)\vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}_i\rangle\Big) = 0 \bmod q'$$

Else:

$$\langle \vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}_i\rangle = \mathbf{0} \bmod q'$$

<u>Form constraints for the $\vec{\boldsymbol{e}}'_i$:</u>
For $i = 1$ to ρ:
For $j = 1$ to n:

If $j \equiv i \bmod \rho$:

$$\mathrm{ct}\Big(\langle \vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}_i\rangle + \langle -\vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}'_i\rangle\Big) = 0 \bmod q'$$

For $k = 1$ to 3:

$$\mathrm{ct}\Big(\langle \sigma_{-1}(X^k)\vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}_i\rangle + \langle \sigma_{-1}(X^{d-k})\vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}'_i\rangle\Big) = 0 \bmod q'$$

(I'm not checking that the other coefficients are 0, it is enough to check that they are 0 in $\vec{\boldsymbol{e}}_i$)
Else:

$$\langle \vec{\boldsymbol{\delta}}_j, \vec{\boldsymbol{e}}'_i\rangle = \mathbf{0} \bmod q'$$

Projection constraints

There will also be 256 constant term constraints per projection that we compute. We compute two projections, using randomly sampled $\mathit{\Pi}_i \in \{0, \pm 1\}^{256\times n}$.

$$\vec{p}_1 = \mathit{\Pi}_1\Big(\sum_{i=1}^{\lceil \frac{n}{\rho}\rceil}\vec{\boldsymbol{y}}_{i,1}\Big) + \mathit{\Pi}_2\Big(\sum_{i=1}^{\lceil \frac{n}{\rho}\rceil}\vec{\boldsymbol{y}}_{i,2}\Big) + \mathit{\Pi}_3\Big(\sum_{i=1}^{\rho}\vec{\boldsymbol{y}}'_{i,1}\Big) + \mathit{\Pi}_4\Big(\sum_{i=1}^{\rho}\vec{\boldsymbol{y}}'_{i,2}\Big) + \mathit{\Pi}_5\Big(\sum_{i=1}^{\lceil \frac{n}{\rho}\rceil}\vec{\boldsymbol{e}}_i\Big) + \mathit{\Pi}_6\Big(\sum_{i=1}^{\rho}\vec{\boldsymbol{e}}'_i\Big)$$
$$\vec{p}_2 = \mathit{\Pi}_7\vec{\boldsymbol{v}}$$

Let $\vec{\boldsymbol{\pi}}_i^{(j)}$ be the j-th row of $\mathit{\Pi}_i$. Then we will have the following projection constraints:

For $j = 1$ to 256:

$$\begin{aligned}&\mathrm{ct}\Big(\sum_{i=1}^{\lceil \frac{n}{\rho}\rceil}\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_1^{(j)}), \vec{\boldsymbol{y}}_{i,1}\rangle + \sum_{i=1}^{\lceil \frac{n}{\rho}\rceil}\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_2^{(j)}), \vec{\boldsymbol{y}}_{i,2}\rangle + \sum_{i=1}^{\rho}\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_3^{(j)}), \vec{\boldsymbol{y}}'_{i,1}\rangle + \sum_{i=1}^{\rho}\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_4^{(j)}), \vec{\boldsymbol{y}}'_{i,2}\rangle\\ &\quad + \sum_{i=1}^{\lceil \frac{n}{\rho}\rceil}\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_5^{(j)}), \vec{\boldsymbol{e}}_i\rangle + \sum_{i=1}^{\rho}\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_6^{(j)}), \vec{\boldsymbol{e}}'_i\rangle - (\vec{p}_1)_j\Big) = 0 \bmod q'\\ &\mathrm{ct}\Big(\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_7^{(j)}), \vec{\boldsymbol{v}}\rangle - (\vec{p}_2)_j\Big) = 0 \bmod q'\end{aligned}$$

The total number of dot product constraints
In total, this gives

$$|\mathcal{F}| = n + 3(\Big\lceil \frac{n}{\rho}\Big\rceil - 1)n + 3(\rho - 1)n \approx 6\sqrt{n}n$$
$$|\mathcal{F}'| = n + dn + (d-4)n + 4n + 512 = (2d+1)n + 512$$

G Coordinate-Wise PSS

Several lattice-based commit-and-open protocols, including LaBRADOR, only reveal an amortized opening to the verifier. With r witness vectors $\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_r$ and some base challenge space S, the verifier sends $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r) \in S^r$ and the prover should respond with $\vec{\boldsymbol{z}} = \sum_{i=1}^{r}\boldsymbol{\alpha}_i\vec{\boldsymbol{w}}_i$. The verifier checks that $\vec{\boldsymbol{z}}$ is a short opening to the amortized commitment. Typically, to extract $\vec{\boldsymbol{w}}_i$, the idea is to obtain two accepting transcripts with amortization challenges that only differ in the i-th coordinate, so that $\boldsymbol{\alpha}_i \neq \boldsymbol{\alpha}'_i$ but $\boldsymbol{\alpha}_j = \boldsymbol{\alpha}'_j$ for all $j \neq i$. If the challenge difference $\boldsymbol{\alpha}_i - \boldsymbol{\alpha}'_i$ is invertible, then a weak opening for the i-th commitment can be computed as $\vec{\boldsymbol{w}}_i = (\boldsymbol{\alpha}_i - \boldsymbol{\alpha}'_i)^{-1}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}')$. Hence, there is a 2-special-sound structure for each of the r coordinates. This is captured by the notion of coordinate-wise special-soundness (CWSS) introduced in [FMN23]. For a recap on CWSS see Appendix B.5.

In this section, we analogously generalize predicate special soundness to coordinate-wise predicate special soundness. The new definitions are merely extensions of the previous ones, setting $r_1 = r_2 = \cdots = r_\mu = 1$ yields regular predicate special soundness. We begin by generalizing our notation for trees of transcripts.

In the coordinate-wise setting, it is hard to define the notation for the predicates without having multiple levels of indexing. To improve readability, we use the convention that $\vec{s}$ denotes vectors over S_m and that $\vec{c}$ denotes vectors over $\mathcal{C}_m = S_m^{r_m}$. Thus, the entries of $\vec{c}$ are themselves vectors $\vec{s}_1, \ldots, \vec{s}_n$. Observe that for all $(\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_\ell) \in \mathcal{C}_m^{(\ell)}$, $\{\vec{s}_1, \vec{s}_{2,1} \ldots \vec{s}_{\ell,r_m}\} \in \mathsf{SS}(S_m, r_m, \ell)$.

Definition G.1 (Coordinate-wise tree of transcripts). *Let $\mu, k_1, \dots, k_\mu, r_1, \dots, r_\mu \in \mathbb{N}_{\geq 0}$ and let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $2\mu+1$-message public-coin argument of knowledge for a relation R_{pp} such that m-th challenge set is $\mathcal{C}_m = S_m^{r_m}$ for some set S_m. Additionally, let $m \in [\mu]$ and $\ell \in [k]$.*

- *We define $\mathbb{T}_{\mu+1}$ to be the set of possible accepting transcripts for Π.*
- *We define $\mathbb{T}_{m+1}^{(\ell)}$ to be the set of possible accepting $(r_1, \dots, r_\mu)$-coordinate-wise $(1, \dots, 1, \ell, k_{m+1}, k_\mu)$-trees of transcripts, and denote $\mathbb{T}_m = \mathbb{T}_{m+1}^{(k_m)}$. We arrange $t \in \mathbb{T}_{m+1}^{(\ell)}$ such that $t = (t_1, \vec{t}_2, \dots, \vec{t}_\ell)$, with $t_1 \in \mathbb{T}_{m+1}$ and $\vec{t}_i \in \mathbb{T}_{m+1}^{r_m}$ for $i > 1$. For any $i \in [r_m]$, the m-th challenges of $t_1, t_{2,i}, \dots, t_{\ell,i}$ in $S_m^{r_m}$ are pairwise distinct in the i-th coordinate and equal in the other coordinates.*
- *For $t \in \mathbb{T}_{m+1}$, we define $\mathsf{trunk}(t)$ to be the prefix $(a_1, c_1, a_2, c_2, \dots, a_m)$ shared by all the transcripts in t, $\mathsf{chal}_m(t)$ to be their shared m-th challenge $\vec{s} \in S_m^{r_m}$, and $\mathsf{chal}_{m,i}(t) = \vec{s}_i$ for $i \in [r_m]$. Furthermore, for $\vec{t} \in \mathbb{T}_{m+1}^{r_m}$ we define $\mathsf{chal}_m(\vec{t})$ to be the vector $\vec{c} = (\vec{s}_1, \dots, \vec{s}_n) \in \mathcal{C}_m^n$ with $\vec{s}_i = \mathsf{chal}_m(t_i)$.*
- *Finally, we let $\mathcal{C}_m^{(\ell)}$ denote the set of tuples $(\vec{s}_1, \vec{c}_2, \dots, \vec{c}_\ell)$ with $\vec{s}_1 \in \mathcal{C}_m$ and $\vec{c}_i = (\vec{s}_{i,1}, \dots, \vec{s}_{i,r_m}) \in \mathcal{C}_m^{r_m}$ such that $\vec{s}_1, \vec{s}_{2,i}, \dots, \vec{s}_{\ell,i}$ are pairwise distinct in the i-th coordinate and equal in the other coordinates. This is the set of all the tuples that may occur as $(\mathsf{chal}_m(t_1), \mathsf{chal}_m(\vec{t}_2), \dots, \mathsf{chal}_m(\vec{t}_\ell))$ for some $(t_1, \vec{t}_2, \dots, \vec{t}_\ell) \in \mathbb{T}_{m+1}^{(\ell)}$.*

For a $(t_1, \vec{t}_2, \dots, \vec{t}_\ell) \in \mathbb{T}_{m+1}^{(\ell)}$ and a coordinate $i \in [r_m]$, we refer to t_1 as the first tree for the i-th coordinate, $t_{2,i}$ as the second tree for that coordinate, $t_{3,i}$ the third, etc. Likewise, for a $(\vec{s}_1, \vec{c}_2, \dots, \vec{c}_\ell) \in \mathcal{C}_m^{(\ell)}$, we refer to $\vec{s}_1$ as the first challenge vector for the i-th coordinate, $\vec{s}_{2,i}$ the second, etc. Furthermore, we say that $s_{1,i}$ is the first challenge for the i-th coordinate, $s_{2,i,i}$ is the second, etc. Because of the asymmetry with the indexing, we will sometimes write $s_{1,i,i}$ to denote $s_{1,i}$ and $t_{1,i}$ to denote t_1.

Definition G.2 (Coordinate-wise predicates). *Let $m \in [\mu]$ and $\ell \in [k_m]$.*

1. *A challenge predicate on level m for the ℓth challenge is a function*

$$\Phi_{m,\ell}^{\mathsf{chal}} : \mathcal{C}_m^{(\ell-1)} \times [r_m] \times S_m \to \{0,1\}.$$

2. *A commitment predicate on level m is a pair $\left(\Phi_m^{\mathsf{prop}}, \Phi_m^{\mathsf{bind}}\right)$, where*

$$\Phi_m^{\mathsf{prop}} : \mathbb{T}_{m+1}^{(k_m-1)} \to \{0,1\} \quad \text{and} \quad \Phi_m^{\mathsf{bind}} : \mathbb{T}_{m+1}^{(k_m-1)} \times [r_m] \times \mathbb{T}_{m+1} \to \{0,1\}.$$

The ℓ-th challenge predicate $\Phi_{m,\ell}^{\mathsf{chal}}$ is now evaluated in every coordinate. In regular predicate special soundness, the ℓ-th challenge predicate ensures that the ℓ-th challenge has the right properties with respect to the previous $\ell - 1$ challenges. A natural coordinate-wise generalization would be to evaluate $\Phi_{m,\ell}^{\mathsf{chal}}$ locally in each coordinate i, giving it as input $s_{1,i}, s_{2,i,i}, \dots, s_{\ell,i,i}$ in the i-th coordinate. However, to increase the expressiveness of the framework, we allow the ℓ-th challenge predicate to take as input the first $\ell - 1$ challenge vectors from *every* coordinate. We do the same for the commitment predicates. The property predicate takes as input the $k_m - 1$ first trees for every coordinate, and then the binding predicate is evaluated in each coordinate i with the k_m-th tree $t_{k_m,i}$. Having information across coordinates was required for us to prove LaBRADOR knowledge sound. We move on to defining the validity of the coordinate-wise subtrees.

Definition G.3 (Coordinate-wise predicate system). *A predicate system $\boldsymbol{\Phi}$ for a $(r_1, \dots, r_\mu)$-coordinate-wise $(k_1, \dots, k_\mu)$-tree structure is a collection of predicates for each level in the tree. The m-th level has one commitment predicate $\left(\Phi_m^{\mathsf{prop}}, \Phi_m^{\mathsf{bind}}\right)$, and k_m challenge predicates $\Phi_{m,1}^{\mathsf{chal}}, \dots, \Phi_{m,k_m}^{\mathsf{chal}}$. We recursively define a series of boolean functions $\boldsymbol{\Phi}_m$ for $m \in [\mu+1]$, describing whether a partial tree of transcripts satisfies the predicate system. For a single accepting transcript $t \in \mathbb{T}_{\mu+1}$ we let $\boldsymbol{\Phi}_{\mu+1}(t) = 1$. For all larger subtrees $t = (t_1, \vec{t}_2, \dots, \vec{t}_{k_m}) \in \mathbb{T}_m$ with $m \in [\mu]$ then $\boldsymbol{\Phi}_m(t) = 1$ if and only if*

$$\begin{aligned} &\bigwedge_{\ell \in [k_m]} \bigwedge_{i \in [r_m]} \left(\boldsymbol{\Phi}_{m+1}(t_{\ell,i}) = 1 \ \wedge \ \Phi_{m,\ell}^{\mathsf{chal}}((\vec{s}_1, \vec{c}_2, \dots, \vec{c}_{\ell-1}), i, s_{\ell,i,i}) = 1\right) \\ &\wedge \Phi_m^{\mathsf{prop}}(t_1, \vec{t}_2, \dots, \vec{t}_{k_m-1}) = 1 \wedge \bigwedge_{i \in [r_m]} \Phi_m^{\mathsf{bind}}((t_1, \vec{t}_2, \dots, \vec{t}_{k_m-1}), i, t_{k_m,i}) = 1, \end{aligned}$$

where $\vec{s}_1 = \mathsf{chal}_m(t_1)$ and $\vec{c}_j = \mathsf{chal}_m(\vec{t}_j)$ for $j > 1$. For notational convenience, we let $\boldsymbol{\Phi} = \boldsymbol{\Phi}_1$.

The definition of coordinate-wise predicate special soundness follows naturally.

Definition G.4 (Coordinate-Wise Predicate Special Soundness). *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $2\mu + 1$-message public-coin argument of knowledge for a relation R_{pp}, with m-th challenge space $\mathcal{C}_m = S_m^{r_m}$. We say that Π is $(\boldsymbol{R}, \boldsymbol{K}, \boldsymbol{\Phi})$-coordinate-wise predicate-special-sound for $\boldsymbol{R} = (r_1, \ldots, r_\mu)$, $\boldsymbol{K} = (k_1, \ldots, k_\mu)$ and a coordinate-wise predicate system $\boldsymbol{\Phi}$ if there exists a polynomial time algorithm which given a statement x and a $\boldsymbol{R}$-coordinate-wise $\boldsymbol{K}$-tree of transcripts t for this statement with $\boldsymbol{\Phi}(t) = 1$ always outputs a witness w such that $(x, w) \in R_{\mathsf{pp}}$.*

To analyze the knowledge soundness of coordinate-wise predicate-special-sound protocols, we define the failure density of our predicates, beginning with the challenge predicates.

Definition G.5 (Failure density of a challenge predicate). *Let $m \in [\mu]$ and $\ell \in [k_m]$. Let $(\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{\ell-1}) \in \mathcal{C}_m^{(\ell-1)}$ be any collection of challenges such that the first $\ell - 1$ challenge predicates are satisfied in every coordinate, meaning*

$$\forall i \in [\ell - 1], \forall j \in [r_m] : \ \Phi^{\mathsf{chal}}_{m,i}((\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{i-1}), j, s_{i,j,j}) = 1.$$

For each coordinate $j \in [r_m]$, consider the set of possible ℓ-th challenges such that $\Phi^{\mathsf{chal}}_{m,\ell}$ fails,

$$B(\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{\ell-1}, j) = \left\{ s \in S_m \ \middle|\ \begin{array}{c} s \notin \{s_{1,j}, s_{2,j,j}, \ldots, s_{\ell-1,j,j}\}, \\ \Phi^{\mathsf{chal}}_{m,\ell}((\vec{c}_1, \ldots, \vec{c}_{\ell-1}), j, s) = 0 \end{array} \right\}.$$

The challenge predicate $\Phi^{\mathsf{chal}}_{m,\ell}$ has failure density $p^{\mathsf{chal}}_{m,\ell}$ if it always holds that $|B(\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{\ell-1})| \leq p^{\mathsf{chal}}_{m,\ell} |S_m|$.

Note that the failure density of the ℓ-th challenge predicate is defined locally for each coordinate. Even though the predicate takes as input the first $\ell - 1$ challenge vectors for every coordinate, we are counting the number of "bad" ℓ-th challenges for a particular coordinate. The same is true for the failure density of commitment predicates.

Definition G.6 (Failure density of a commitment predicate). *Let $m \in [\mu]$. Define a set of bad subtrees*

$$\mathsf{Bad}^{\mathsf{prop}}_m = \left\{ (t_1, \vec{t}_2, \ldots, \vec{t}_{k_m - 1}) \in \mathbb{T}^{(k_m-1)}_{m+1} \ \middle|\ \begin{array}{c} \forall i \in [k_m - 1], j \in [r_m] : \\ \boldsymbol{\Phi}_{m+1}(t_{i,j}) = 1, \\ \Phi^{\mathsf{chal}}_{m,i}(\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{i-1}, j, s_{i,j,j}) = 1, \\ \Phi^{\mathsf{prop}}_m(t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}) = 0 \end{array} \right\}$$

using the shorthand $\vec{s}_1 = \mathsf{chal}_m(t_1)$ and $\vec{c}_i = \mathsf{chal}_m(\vec{t}_i)$ for $i > 1$. That is, subtrees in $\mathsf{Bad}^{\mathsf{prop}}_m$ fail to satisfy the property predicate but otherwise satisfy the constraints. For each $(t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}) \in \mathbb{T}^{(k_m-1)}_{m+1}$, we define $\mathsf{Sat}^{\mathsf{bind}}_m(t_1, \ldots, t_{k_m-1})$ to be the set of possible k_m-th subtrees that satisfy the binding predicate and the other constraints, using the shorthand $s = \mathsf{chal}_{m,j}(t)$.

$$\mathsf{Sat}^{\mathsf{bind}}_m(t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}, j) = \left\{ t \in \mathbb{T}_{m+1} \ \middle|\ \begin{array}{c} \forall i \in [k_m - 1] : \ s \neq s_{i,j,j}, \\ \mathsf{trunk}(t) = \mathsf{trunk}(t_1), \\ \boldsymbol{\Phi}_{m+1}(t) = 1, \\ \Phi^{\mathsf{chal}}_{m,k_m}((\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{k_m-1}), j, s) = 1 \\ \Phi^{\mathsf{bind}}_m((t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}), j, t) = 1 \end{array} \right\}$$

Consider the m-th level challenges for the j-th coordinate occuring for some tree in this set,

$$B(t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}, j) = \left\{ \mathsf{chal}_{m,j}(t) \ \middle|\ t \in \mathsf{Sat}^{\mathsf{bind}}_m(t_1, \vec{t}_2, \ldots, t_{k_m-1}, j) \right\}.$$

The commitment predicate $(\Phi^{\mathsf{prop}}_m, \Phi^{\mathsf{bind}}_m)$ has failure density p^{com}_m if it always holds that there is some coordinate $j \in [r_m]$ such that

$$|B(t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}, j)| \leq p^{\mathsf{com}}_m |S_m|.$$

In coordinate-wise special-sound protocols, one witness element $\vec{\boldsymbol{w}}_i$ can typically be extracted from every coordinate i. When the property predicate does not hold, we require that there is some coordinate where we can apply the failure density for the binding predicate. In other words, there must be some coordinate i where only a small fraction of challenges allow the prover to use the same witness $\vec{\boldsymbol{w}}_i$, and thereby not violate binding.

When the binding predicate is not satisfied in some coordinate, we must be able to obtain a witness for the binding relation.

Definition G.7 (Coordinate-wise binding relation). *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $(\boldsymbol{R}, \boldsymbol{K}, \boldsymbol{\Phi})$-coordinate-wise predicate-special-sound argument of knowledge for a relation R_{pp}, and let $R_{\mathsf{bind,pp}}$ be an additional relation. We say that $\boldsymbol{\Phi}$ admits R_{bind} as a coordinate-wise binding relation if there exists a polynomial time algorithm $\mathcal{B}$, with the following property. Say it gets as input some $(t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}) \in \mathbb{T}_{m+1}^{(k_m-1)}$, $j \in [r_m]$ and $t \in \mathbb{T}_{m+1}$ such that*

$$\Phi_m^{\mathsf{bind}}((t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}), j, t) = 0 \text{ and}$$
$$\forall \ell \in [k_m], i \in [r_m]: \ \boldsymbol{\Phi}_{m+1}(t_{\ell,i}) = 1 \text{ and } \Phi_{m,\ell}^{\mathsf{chal}}((\vec{s}_1, \vec{c}_2, \ldots, \vec{c}_{\ell-1}), i, s_{\ell,i,i}) = 1,$$

where $\vec{s}_1 = \mathsf{chal}_m(t_1)$ and $\vec{c}_i = \mathsf{chal}_m(\vec{t}_i)$ for $i > 1$. Then it always holds that $\mathcal{B}((t_1, \vec{t}_2, \ldots, \vec{t}_{k_m-1}), j, t) \in R_{\mathsf{bind,pp}}$.

We may now finally describe the knowledge soundness of a coordianate-wise predicate-special-sound protocol.

Theorem G.1. *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $(\boldsymbol{R}, \boldsymbol{K}, \boldsymbol{\Phi})$-coordinate-wise predicate-special-sound argument of knowledge for a relation R. In addition, let R_{bind} be a coordinate-wise binding relation for $\boldsymbol{\Phi}$. Then the adaptive Fiat-Shamir transformation $FS[\Pi]$ is adaptively knowledge sound for the relation $R_{\mathsf{pp}} \cup R_{\mathsf{bind,pp}}$ with knowledge error*

$$2(Q+1)\sum_{i=1}^{\mu} r_i \cdot \max\left(\frac{k_i - 1}{|\mathcal{C}_i|}, p_i^{\mathsf{com}} + \sum_{\ell=1}^{k_i} p_{i,\ell}^{\mathsf{chal}}\right).$$

The number of times that the knowledge extractor invokes the prover is in expectation at most $K + Q(K-1)$, where $K = \prod_{i=1}^{\mu}(r_i(k_i - 1) + 1)$.

H Knowledge Soundness Proof for PSS

In this section, we present our proof for Theorem 4.1 and Theorem G.1. First, we show that for the adaptive Fiat-Shamir transformation of any predicate-special-sound protocol, we can construct an efficient knowledge extractor that outputs a witness with knowledge error depending only on the tree structure and the failure densities of the predicates. To this end, our approach is to extend the seminal work [AFK22] by Attema, Fehr and Klooß, which analyzes the knowledge soundness of multi-round special-sound protocols. Conceptually, our knowledge extractor is the same as theirs, except for some additional predicate checks. These predicate checks do not affect the expected runtime of the extractor, only its success probability. To capture the core of their extraction strategy, [AFK22] introduces an abstract sampling game. Analyzing the properties of this abstract game allows them to derive the knowledge error and expected runtime of their knowledge extractor without being bogged down with heavy notation. We follow their approach.

Next, we generalize our extractor construction to coordinate-wise predicate-special-sound protocols. The extractor for coordinate-wise special-sound protocols in [FMN23] essentially runs the [AFK22] extractor for special sound protocols in each coordinate. Our extractor is the [FMN23] extractor extended with predicate checks at the very end.

The rest of this section is organized as follows. In Section H.1, we present and analyze a new abstract sampling game for our predicate special soundness knowledge extractor. Then in Section H.2, we describe the knowledge extractor and analyze its efficiency and success probability using the abstract game. Finally in Section H.3, we generalize our analysis to coordinate-wise predicate special soundness.

H.1 Abstract Sampling Game

The abstract sampling game is presented in Game 3. Entries are sampled from the array M exactly like in [AFK22]. The only changes are that each entry of M has an extra t element and that we might additionally fail at the very end because of the predicate functions.

Before analyzing the properties of the abstract sampling game, we will define some helpful notation. Following [AFK22], for all $i \in [U]$ we define the function

$$a_i : ([N]^r)^U \to \mathbb{N}_{\geq 0},$$
$$(\vec{j}_1, \ldots, \vec{j}_U) \mapsto \left|\left\{\vec{j} \in [N]^r \;\middle|\; M(\vec{j}_1, \ldots, \vec{j}_{i-1}, \vec{j}, \vec{j}_{i+1}, \ldots, \vec{j}_U) = (1, i, \cdot)\right\}\right|.$$

Game 3: The abstract sampling game with predicates

Parameters:

- $k, N, U \in \mathbb{N}$ and a set T.
- U-dimensional array M with entries $M(j_1, \ldots, j_U) \in \{0,1\} \times [U] \times T$ for all $(j_1, \ldots, j_U) \in [N]^U$.
- Functions $f_\ell^{\mathsf{chal}} : [N]^\ell \to \{0,1\}$ for $\ell \in [k]$.
- A pair of functions $(f^{\mathsf{prop}}, f^{\mathsf{bind}})$ with $f^{\mathsf{prop}} : T^{k-1} \to \{0,1\}$ and $f^{\mathsf{bind}} : T^k \to \{0,1\}$.

Game:

1. Sample $(j_1, \ldots, j_U) \xleftarrow{\$} [N]^U$ and compute $(v, i, t) = M(j_1, \ldots, j_U)$.
2. If $v = 0$, abort with output 0.
3. Set $c_1 \coloneqq j_i$, $t_1 \coloneqq t$, $\ell \coloneqq 1$ and $W \coloneqq [N] \setminus \{c_1\}$.
4. Repeat the following until $\ell = k$ or $W = \varnothing$:
   (a) Sample $j' \xleftarrow{\$} W$ and set $W \coloneqq W \setminus \{j'\}$.
   (b) Compute $(v', i', t') = M(j_1, \ldots, j_{i-1}, j', j_{i+1}, \ldots, j_U)$.
   (c) If $v' = 1$ and $i' = i$, set $\ell \coloneqq \ell + 1$, $c_\ell \coloneqq j'$ and $t_\ell \coloneqq t'$.
5. If $\ell < k$ or $f_n^{\mathsf{chal}}(c_1, \ldots, c_n) = 0$ for some $n \in [k]$, output 0.
6. Output $f^{\mathsf{prop}}(t_1, \ldots, t_{k-1}) \vee \neg f^{\mathsf{bind}}(t_1, \ldots, t_k)$.

This function counts the number of entries with $v = 1$ and index i in the 1-dimensional array $M(j_1, \ldots, j_{i-1}, \cdot, j_{i+1}, \ldots, j_U)$. Notice that the function does not depend on the i-th input j_i. Therefore we sometimes, by a slight abuse of notation, write $a_i(j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_U)$.

Lemma H.1. *Consider the game in Game 3. Let $\vec{J} = (J_1, \ldots, J_U)$ be uniformly distributed in $[N]^U$, indicating the first entry sampled, and let $(V, I, T) = M(\vec{J})$. For each $i \in [U]$, let $A_i = a_i(\vec{J})$. Let F_ℓ^{chal} be the random variable indicating the outcome of f_ℓ^{chal}. Let F^{prop} and F^{bind} be the random variables for the outcome of f^{prop} and f^{bind} respectively. Let $d_1^{\mathsf{chal}}, \ldots, d_k^{\mathsf{chal}}, d^{\mathsf{com}} \in [0,1]$ be numbers such that for all $i \in [U]$ and all $a \in \mathbb{N}$, $k \le a \le N$,*

$$\frac{d_\ell^{\mathsf{chal}} N}{a - \ell + 1} \ge \Pr\left[F_\ell^{\mathsf{chal}} = 0 \;\middle|\; V = 1, I = i, A_i = a, \bigcap_{n=1}^{\ell-1} F_n^{\mathsf{chal}} = 1\right] \quad \text{and}$$

$$\frac{d^{\mathsf{com}} N}{a - k + 1} \ge \Pr\left[F_k^{\mathsf{chal}} = 1, F^{\mathsf{bind}} = 1 \;\middle|\; \begin{array}{r} V = 1, I = i, A_i = a, \\ \bigcap_{\ell=1}^{k-1} F_\ell^{\mathsf{chal}} = 1, F^{\mathsf{prop}} = 0 \end{array}\right].$$

Finally, set some tuning parameter $1 \le \sigma \le N/k$ such that $\sigma k \in \mathbb{N}$. Then the game ouputs 1 with probability at least

$$\Pr\left[V = 1\right] - A \cdot \max\left(\frac{\sigma k - 1}{N}, \frac{\sigma k}{\sigma k - k + 1} d^{\mathsf{com}} + \sum_{\ell=1}^{k} \frac{\sigma k}{\sigma k - \ell + 1} d_\ell^{\mathsf{chal}}\right),$$

where $A = \sum_{i=1}^{U} \Pr\left[A_i > 0\right]$.

Proof. By the description of the abstract game, we have that the probability that the game outputs 1 is

$$\sum_{i=1}^{U} \Pr\left[V = 1, I = i, A_i \ge k, \bigcap_{\ell=1}^{k} F_\ell^{\mathsf{chal}} = 1, (F^{\mathsf{prop}} = 1 \cup F^{\mathsf{bind}} = 0)\right]$$
$$\ge \sum_{i=1}^{U} \Pr\left[V = 1, I = i, A_i \ge \sigma k, \bigcap_{\ell=1}^{k} F_\ell^{\mathsf{chal}} = 1, (F^{\mathsf{prop}} = 1 \cup F^{\mathsf{bind}} = 0)\right].$$

We lower the success probability to allow bounding the impact of the failure density when sampling without replacement.

$$
\begin{aligned}
&= \Pr\left[V=1\right] - \sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i < \sigma k\right] && (9)\\
&- \sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i \geq \sigma k, \bigcup_{\ell=1}^{k} F_\ell^{\mathsf{chal}} = 0\right] && (10)\\
&- \sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i \geq \sigma k, \bigcap_{\ell=1}^{k} F_\ell^{\mathsf{chal}} = 1, F^{\mathsf{prop}} = 0, F^{\mathsf{bind}} = 1\right] && (11)
\end{aligned}
$$

The equality holds by repeated application of $\Pr\left[A \cap B\right] = \Pr\left[A\right] - \Pr\left[A \cap \neg B\right]$ for events A, B. The expression contains three sums that represent each of the failure cases. We will individually upper bound each case, starting with (9).

$$
\begin{aligned}
\sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i < \sigma k\right] &= \sum_{i=1}^{U} \sum_{a=0}^{\sigma k-1} \Pr\left[A_i = a\right] \Pr\left[V=1, I=i \mid A_i = a\right]\\
&= \sum_{i=1}^{U} \sum_{a=0}^{\sigma k-1} \Pr\left[A_i = a\right] \frac{a}{N}\\
&\leq \sum_{i=1}^{U} \sum_{a=1}^{\sigma k-1} \Pr\left[A_i = a\right] \frac{\sigma k - 1}{N}
\end{aligned}
$$

Next, we upper bound the second sum (10). For the sake of conciseness, we let $E_{i,a}$ denote the event that $V=1, I=i, A_i=a$.

$$
\begin{aligned}
&\sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i \geq \sigma k, \bigcup_{\ell=1}^{k} F_\ell^{\mathsf{chal}} = 0\right]\\
&= \sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i \geq \sigma k, \bigcup_{\ell=1}^{k} \left(F_\ell^{\mathsf{chal}} = 0, \bigcap_{n=1}^{\ell-1} F_n^{\mathsf{chal}} = 1\right)\right]\\
&\leq \sum_{i=1}^{U} \sum_{\ell=1}^{k} \Pr\left[V=1, I=i, A_i \geq \sigma k, F_\ell^{\mathsf{chal}} = 0, \bigcap_{n=1}^{\ell-1} F_n^{\mathsf{chal}} = 1\right]\\
&= \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \frac{a}{N} \sum_{\ell=1}^{k} \Pr\left[F_\ell^{\mathsf{chal}} = 0, \bigcap_{n=1}^{\ell-1} F_n^{\mathsf{chal}} = 1 \;\middle|\; E_{i,a}\right]\\
&\leq \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \frac{a}{N} \sum_{\ell=1}^{k} \Pr\left[F_\ell^{\mathsf{chal}} = 0 \;\middle|\; E_{i,a}, \bigcap_{n=1}^{\ell-1} F_n^{\mathsf{chal}} = 1\right]\\
&\leq \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \frac{a}{N} \sum_{\ell=1}^{k} \frac{d_\ell^{\mathsf{chal}} N}{a - \ell + 1}\\
&= \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \sum_{\ell=1}^{k} \frac{a}{a - \ell + 1} d_\ell^{\mathsf{chal}}\\
&\leq \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \sum_{\ell=1}^{k} \frac{\sigma k}{\sigma k - \ell + 1} d_\ell^{\mathsf{chal}}
\end{aligned}
$$

For the last step, we exploited that $a \geq \sigma k \geq 1$ implies $a/(a - \ell + 1) \leq \sigma k/(\sigma k - \ell + 1)$. We will use a very similar approach to upper bound the third sum (11). Recall that $F^{\mathsf{prop}} = 0$ is determined by the first

$k-1$ successes sampled.

$$\begin{aligned}
&\sum_{i=1}^{U} \Pr\left[V=1, I=i, A_i \geq \sigma k, \bigcap_{\ell=1}^{k} F_\ell^{\mathsf{chal}}=1, F^{\mathsf{prop}}=0, F^{\mathsf{bind}}=1\right] \\
&\leq \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \frac{a}{N} \\
&\quad \cdot \Pr\left[F_k^{\mathsf{chal}}=1, F^{\mathsf{bind}}=1 \,\middle|\, E_{i,a}, \bigcap_{\ell=1}^{k-1} F_\ell^{\mathsf{chal}}=1, F^{\mathsf{prop}}=0\right] \\
&\leq \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \frac{a}{N} \cdot \frac{d^{\mathsf{com}} N}{a-k+1} \\
&\leq \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \frac{\sigma k}{\sigma k - k + 1} d^{\mathsf{com}}
\end{aligned}$$

Using the bounds for the three sums (9,10,11), the lemma follows.

$$\begin{aligned}
&\Pr\left[V=1\right] - \sum_{i=1}^{U} \sum_{a=1}^{\sigma k-1} \Pr\left[A_i = a\right] \frac{\sigma k - 1}{N} \\
&\qquad - \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_i = a\right] \left(\frac{\sigma k}{\sigma k - k + 1} d^{\mathsf{com}} + \sum_{\ell=1}^{k} \frac{\sigma k}{\sigma k - \ell + 1} d_\ell^{\mathsf{chal}}\right) \\
&\geq \Pr\left[V=1\right] - A \cdot \max\left(\frac{\sigma k - 1}{N}, \frac{\sigma k}{\sigma k - k + 1} d^{\mathsf{com}} + \sum_{\ell=1}^{k} \frac{\sigma k}{\sigma k - \ell + 1} d_\ell^{\mathsf{chal}}\right)
\end{aligned}$$

□

The rest of the analysis of the abstract sampling game is identical to the one in [AFK22]. For completeness, we include the relevant results below.

Lemma H.2 (Expected cost [AFK22, Lemma 5]). *Consider the game in Game 3, as well as a cost function $\varGamma : [N]^U \to \mathbb{R}_{\geq 0}$ and a constant cost $\gamma \in \mathbb{R}_{\geq 0}$. Let $\vec{J} = (J_1, \dots J_U)$ be uniformly distributed in $[N]^U$, indicating the first entry sampled, and let $(V, I, T) = M(\vec{J})$. For each $i \in [U]$, let $A_i = a_i(\vec{J})$. We define the cost of sampling an entry $M(j_1, \dots, j_U) = (v, i, t)$ with index $i = I$ to be $\varGamma(j_1, \dots, j_U)$ and the cost of sampling an entry $M(j_1, \dots, j_U) = (v, i, t)$ with $i \neq I$ to be γ. Let $\varDelta$ be the total cost of playing this game. Then*

$$\mathbb{E}[\varDelta] \leq k \cdot \mathbb{E}[\varGamma(\vec{J})] + (k-1) \cdot A' \cdot \gamma,$$

where $A' = \sum_{i=1}^{U} \Pr\left[I \neq i, A_i > 0\right] \leq A$.

Lemma H.3 ([AFK22, Lemma 3 & 6]). *Consider the game in Game 3. Let v, idx and $tree$ be functions such that $M(\vec{j}) = (v(\vec{j}), idx(\vec{j}), tree(\vec{j}))$ for all $\vec{j} \in [N]^U$. Furthermore, let $\vec{J} = (J_1, \dots, J_U)$ be uniformly distributed in $[N]^U$ and set $A_i = a_i(J)$ for all $i \in [U]$. Assume that for all $\vec{j} \in [N]^U$ there exists a subset $S(\vec{j}) \subseteq [U]$ of cardinality at most Q such that $idx(\vec{j}) = idx(\vec{j}')$ for all $\vec{j}, \vec{j}'$ with $j_\ell = j'_\ell$ for all $\ell \in S(\vec{j})$. Then*

$$\begin{aligned}
A &= \sum_{i=1}^{U} \Pr\left[A_i > 0\right] \leq Q + 1 \text{ and} \\
A' &= \sum_{i=1}^{U} \Pr\left[idx(J) \neq i, A_i > 0\right] \leq Q.
\end{aligned}$$

H.2 Knowledge Extractor

We move on to discuss how to construct a knowledge extractor for the adaptive Fiat-Shamir transformation $\mathsf{FS}[\varPi]$ of some $(\boldsymbol{K}, \boldsymbol{\varPhi})$-predicate-special-sound interactive argument $\varPi$. To simplify the presentation, like [AFK22] we will only present the case where all the rounds of Π use the same challenge set $\mathcal{C}$ of cardinality N. The analysis can easily be generalized to the case where each round i has a different challenge set $\mathcal{C}_i$ of cardinality N_i. The proof is the same as before, except that we need to handle the additional bookkeeping from having a differen random oracle $\mathsf{RO}_i : \{0,1\}^{\leq u} \to \mathcal{C}_i$ for each round.

Let $\mathcal{P}^*$ be some adaptive Q-query random oracle prover for $\mathsf{FS}[\Pi]$. By Remark B.1, we can assume that $\mathcal{P}^*$ is deterministic. After making at most Q queries to the random oracle, the prover outputs $(x, \pi, \mathsf{aux}) \leftarrow \mathcal{P}^*$, where x is a statement, $\pi = (a_1, \ldots, a_{\mu+1})$ a proof and aux some auxilliary information. We reformat the output of $\mathcal{P}^*$ to be an index vector $\vec{I} = (I_1, \ldots, I_{\mu+1})$, where

$$I_1 \coloneqq (x, a_1), I_2 \coloneqq (x, a_1, a_2), \ldots, I_{\mu+1} \coloneqq (x, a_1, \ldots, a_{\mu+1}).$$

On top of that, we extend $\mathcal{P}^*$ to a $(Q+\mu)$-query algorithm $\mathcal{A}$ that checks the validity of the proof. It computes $\boldsymbol{c}_i = \mathsf{RO}(I_i)$ for $i \in [\mu]$ and outputs

$$\vec{I}, t \coloneqq (a_1, c_1, \ldots, a_\mu, c_\mu, a_{\mu+1}), v \coloneqq \mathcal{V}(x, t), b \coloneqq 1 \text{ and } \mathsf{aux}.$$

The bit b indicates whether t satisfies the binding constraints or not. For $\mathcal{A}$ we trivially have that $b = 1$, but we include it to make the output of $\mathcal{A}$ consistent with our extractor.

The extractor is constructed recursively as a sequence of subextractors $\mathcal{E}_1, \ldots, \mathcal{E}_{\mu+1}$. The goal of $\mathcal{E}_m$ is to output a tree $t \in \mathbb{T}_m$ such that either $\boldsymbol{\Phi}(t) = 1$ or such that t encodes a witness for the binding relation R_{bind}. To do so, it essentially plays an instantiation of the abstract sampling game with the previous subextractor $\mathcal{E}_{m+1}$. At the bottom of the recursion, $\mathcal{E}_{\mu+1} = \mathcal{A}$. For the recursive argument to work, we need the subextractors to be the same kind of object as $\mathcal{A}$, namely a random oracle algorithm making the same number of queries. We define $\mathcal{E}_m$ in Extractor 1. The final extractor $\mathcal{E}$ for $\mathsf{FS}[\Pi]$ is obtained by running $\mathcal{E}_1$ and using lazy-sampling to answer its random oracle queries.

Extractor 1: The subextractor $\mathcal{E}_m$

Parameters:

- $k_m, Q \in \mathbb{N}$.
- Challenge predicates $\Phi^{\mathsf{chal}}_{m,\ell} : \mathcal{C}^\ell \to \{0,1\}$ for $\ell \in [k_m]$.
- A commitment predicate $(\Phi^{\mathsf{prop}}_m, \Phi^{\mathsf{bind}}_m)$, where $\Phi^{\mathsf{prop}}_m : \mathbb{T}_{m+1}^{(k_m-1)} \to \{0,1\}$ and $\Phi^{\mathsf{bind}}_m : \mathbb{T}_{m+1}^{(k_m-1)} \times \mathbb{T}_{m+1} \to \{0,1\}$.

Black-box access to: $\mathcal{E}_{m+1}$.
Random oracle queries: $Q + \mu$.

1. Run $\mathcal{E}_{m+1}$ as follows to obtain $(\vec{I}, t, v, b, \mathsf{aux})$: Relay the $Q+\mu$ queries to the random oracle and record all query-response pairs. Set $i \coloneqq I_m$ and let $j_i \in \mathcal{C}$ be the response to query i.
2. If $v = 0$, the extractor fails with output $v \coloneqq 1$.
3. Set $b_1 \coloneqq b$, $c_1 \coloneqq j_i$, $t_1 \coloneqq t$, $W \coloneqq \mathcal{C} \setminus \{c_1\}$ and $\ell \coloneqq 1$.
4. Repeat the following until $\ell = k_m$ or $W = \varnothing$:
   (a) Sample $j' \xleftarrow{\$} W$ and set $W \coloneqq W \setminus \{j'\}$.
   (b) Run $\mathcal{E}_{m+1}$ as follows to obtain $(\vec{I}', t', v', b', \mathsf{aux}')$:
      i. Stop after the initial run of $\mathcal{P}^*$ if $I'_m \neq I_m$.
      ii. Answer the query to i with j'.
      iii. Answer other queries from $\mathcal{E}_{m+1}$ consistently with the responses stored. Each new query is answered with a locally sampled fresh random value from $\mathcal{C}$, and recorded.
   (c) If $v' = 1$ and $I'_m = I_m$, set $\ell \coloneqq \ell + 1$, $b_\ell = b'$, $c_\ell \coloneqq j'$ and $t_\ell \coloneqq t'$.
5. If $\ell < k_m$ or $\Phi^{\mathsf{chal}}_{m,n}(c_1, \ldots, c_n) = 0$ for some $n \in [k_m]$, the extractor fails with output $v \coloneqq 0$.
6. If $b_n = 0$ for some $n \in [k_m]$ or $\Phi^{\mathsf{bind}}_m((t_1, \ldots, t_{k_m-1}), t_{k_m}) = 0$, let $b \coloneqq 0$. Else, let $b \coloneqq 1$.
7. If $\Phi^{\mathsf{prop}}_m(t_1, \ldots, t_{k_m-1}) = 0$ and $b = 1$, the extractor fails with output $v \coloneqq 0$. Else, it succeeds with output $\vec{I}, (t_1, \ldots, t_{k_m}), v \coloneqq 1, b, \mathsf{aux}$.

For the sake of a simpler description of $\mathcal{E}_m$ that is more consistent with the abstract sampling game, $\mathcal{E}_m$ does not terminate immediately when $\mathcal{E}_{m+1}$ outputs a tuple with $v = 1$ and $b = 0$, even though it has found a valid witness for R_{bind}. Note that this optimization would not lead to an improved result for the success probability or the expected run time, as in the worst case we always have that $\mathcal{E}_{m+1}$ outputs $b = 1$.

For adaptive security, the statement is considered part of the first prover message a_1. If the extractor $\mathcal{E}$ is able to output a tree of transcripts, it holds immediately that they all have the same statement. To fit Definition B.11, the output format of $\mathcal{E}$ should be modified to the form $(x, \pi, \mathsf{aux}, v, w)$, where (x, π, aux) is identically distributed to the output of $\mathcal{P}^*$ and $v = \mathcal{V}(x, \pi)$. This change can easily be implemented in Extractor 1, but we omit it to make the presentation more clear.

Lemma H.4 (Correctness and consistency [AFK22, Lemma 7 & Proposition 1]). *For any fixed choice of the random oracle* RO, *let* $(\vec{I}, (t^{(1)}, \dots, t^{(k_m)}), v, b) \leftarrow \mathcal{E}_m^{\mathsf{RO}}$. *If* $v = 1$, *then* $(t^{(1)}, \dots, t^{(k_m)}) \in \mathbb{T}_m$. *Furthermore, the index vector* $\vec{I}$ *and the auxilliary information* aux *equal those output by* $\mathcal{P}^*$ *quering* RO.

The subextractor $\mathcal{E}_m$ samples entries in exactly the same way as in [AFK22], but it might additionally fail because of the predicates. Hence, we can use the same argument for the expected run time of $\mathcal{E}_m$ as in [AFK22], but the success probability analysis now relies on Lemma H.1.

Lemma H.5 (Run Time and Success Probability). *Let* $N = |\mathcal{C}|$. *The extractor* $\mathcal{E}_m$ *succeeds and outputs* $v = 1$ *with probability at least*

$$\varepsilon(\mathcal{P}^*) - (Q+1) \sum_{i=\mu}^{m} \min_{\sigma_i \in [1, \frac{N}{k_i}]:\ \sigma_i k_i \in \mathbb{N}} \kappa_i(\sigma_i),$$

where

$$\kappa_i(\sigma) = \max\left(\frac{\sigma k_i - 1}{N}, \frac{\sigma k_i}{\sigma k_i - k_i + 1} p_i^{\mathsf{com}} + \sum_{\ell=1}^{k_i} \frac{\sigma k_i}{\sigma k_i - \ell + 1} p_{i,\ell}^{\mathsf{chal}}\right).$$

The number of times it invokes $\mathcal{P}^*$ *is in expectation at most* $K_m + Q(K_m - 1)$, *where* $K_m = \prod_{i=m}^{\mu} k_i$.

Proof. The proof is by induction in m. The base case $m = \mu + 1$ holds trivially. For the induction step, we assume that the lemma holds for $m+1$. The idea of the proof is to demonstrate how one can view $\mathcal{E}_m$ as playing an instantiation of the abstract sampling game in Game 3 with $\mathcal{E}_{m+1}$, allowing us to derive the properties of $\mathcal{E}_m$ from the properties of the game.

Let $U = |\{0,1\}^{\le u}|$ denote the cardinality of the domain of the random oracle $\mathsf{RO} : \{0,1\}^{\le u} \to \{0,1\}$. Fix an arbitrary ordering $\xi_1, \dots, \xi_U$ for the bitstrings $\xi_i \in \{0,1\}$. Then a vector $\vec{j} \in \mathcal{C}^U$ encodes the function table of the random oracle with $\mathsf{RO}(\xi_i) = j_i$. Within a run of $\mathcal{E}_m$, all queries made by the different invocations of $\mathcal{E}_{m+1}$ are answered consistently using lazy sampling, except for the queries to the index i, where we reprogram the response. This is indistinguishable from how it is done in the abstract sampling game, where the entire function table $\vec{j}$ of the random oracle is sampled initially. For the purpose of analysis, we modify $\mathcal{E}_m$ to handle the random oracle queries like in the game. This change has no impact on its success probability or its expected number of prover invocations.

Next, let us define the array M. Observe that since $\mathcal{A}$ is deterministic, we can view it as a function from a choice of random oracle to its output given access to that random oracle. Then for $\mathcal{E}_\mu$, we can define $M(\vec{\boldsymbol{c}}) = (v, I_\mu, t)$, where $(\vec{I}, t, v, b, \mathsf{aux}) \leftarrow \mathcal{A}(\vec{\boldsymbol{c}})$. However, for $\mathcal{E}_m$, the subextractor it invokes $\mathcal{E}_{m+1}$ is not a deterministic algorithm. Defining the entries of M by the output of $\mathcal{E}_{m+1}$ will not give us a deterministic array like in the game.

The trick to still be able to use the abstract game is to analyze the case when $\mathcal{E}_{m+1}$ is using some fixed random tape. In this case, M is a deterministic array like in the sampling game. Averaging over the choice of random tape for $\mathcal{E}_{m+1}$ will, by the linearity of the success probability and the expected run time, yield the desired result. Formally, to allow for fresh randomness in the different runs of $\mathcal{E}_{m+1}$ within $\mathcal{E}_m$, we actually fix a choice of function $f : \mathcal{C}^U \to \{0,1\}^{\le r}$. The invocation of $\mathcal{E}_{m+1}$ with random oracle $\vec{j} \in \mathcal{C}^U$ will use $f(\vec{j})$ as its random tape.

We instaniate f_ℓ^{chal} with $\Phi_{m,\ell}^{\mathsf{chal}}$ for $\ell \in [k_m]$ and $(f^{\mathsf{prop}}, f^{\mathsf{bind}})$ with $(\Phi_m^{\mathsf{prop}}, \Phi_m^{\mathsf{bind}})$. A difference between $\mathcal{E}_m$ and the abstract game is when the predicates lead to failure. Namely, the difference is the b bits output by $\mathcal{E}_{m+1}$, which are not present in the abstract game. If $b_n = 0$ for some $n \in [k]$, then $\mathcal{E}_m$ will succeed even though $\Phi_m^{\mathsf{prop}}(t_1, \dots, t_{k_m - 1})$ and $\Phi_m^{\mathsf{bind}}((t_1, \dots, t_{k_m-1}), t_{k_m}) = 1$, so that it would fail in the game. Luckily, we are lower bounding the success probability of $\mathcal{E}_m$, so we will simply assume that we always have $b_n = 1$ for all $n \in [k_m]$. Clearly, this can only decrease the success probability of $\mathcal{E}_m$. As a consequence, the b output of $\mathcal{E}_{m+1}$ can be ignored completely, so that $\mathcal{E}_m$ is truly playing the abstract game.

Next, we need to bound the parameters $A = \sum_{i=1}^{U} \Pr\left[A_i > 0\right]$ and $d_1^{\mathsf{chal}}, \dots d_k^{\mathsf{chal}}, d^{\mathsf{com}}$ from Lemma H.1. We bound A using Lemma H.3. The observation is that the set $S(\vec{\boldsymbol{c}}) \subseteq \{0,1\}^{\le u}$ corresponds to the set of at most Q indices that the prover queries to the random oracle when the random oracle is $\vec{\boldsymbol{c}}$. Since $\mathcal{P}^*$ is deterministic, its output can only change when the random oracle is reprogrammed at one of the indices in $S(\vec{\boldsymbol{c}})$. This then also holds for the index vector output by $\mathcal{E}_{m+1}$, since $\mathcal{E}_{m+1}$ outputs the same index vector $\vec{I}$ as $\mathcal{P}^*$ for any random oracle $\vec{c}$ by Lemma H.4. It follows that we can apply Lemma H.3 to bound $A \leq Q+1$.

We can use Definition 4.5 to bound $d_1^{\mathsf{chal}}, \ldots, d_{k_m}^{\mathsf{chal}}$. The first observation is that if $\mathcal{E}_{m+1}$ outputs an entry with $v = 1$, $b = 1$ and subtree $t \in \mathbb{T}_{m+1}$, then by construction $\boldsymbol{\Phi}_{m+1}(t) = 1$. Next, given that $\mathcal{E}_m$ has obtained a first success with index i and is sampling from a 1-dimensional array with $a \geq k_m$, it will obtain a uniform random size k_m subset of these a successes. Fix any choice for $c_1, \ldots, c_{\ell-1}$ such that the first $\ell - 1$ challenge predicates are satisfied. Then we know that the ℓ-th challenge is uniformly distributed among the $a - \ell + 1$ remaining successes. Out of these, at most $p_{m,\ell}^{\mathsf{chal}} N$ are such that $\Phi_\ell^{\mathsf{chal}}$ fails. Thus, we can set $d_\ell^{\mathsf{chal}} = p_\ell^{\mathsf{chal}}$ for all $\ell \in [k_m]$.

Similarly, we can use Definition 4.6 to set $d^{\mathsf{com}} = p_m^{\mathsf{com}}$. Fix any choice for the first $k-1$ successes such that the challenge predicates are satisfied, but the property predicate is not. Then the k_m-th challenge is uniformly distributed among the remaining $a - k_m + 1$ successes. Out of these, we know that there are at most $p_m^{\mathsf{com}} N$ challenges such that the k_m-th challenge predicate and the binding predicate are satisfied. Thus, we can set $d^{\mathsf{com}} = p_m^{\mathsf{com}}$.

Let us now derive the lower bound for the success probability of $\mathcal{E}_m$. Let F be the random variable indicating the choice of random tape function, let $\vec{J}$ denote the initial choice of random oracle in $\mathcal{C}^U$ and let $S_{m'}$ for $n \in \{m, m+1\}$ be the event that $\mathcal{E}_n$ outputs $v = 1$ when $\mathcal{E}_{m+1}$ is run with the random oracle $\vec{J}$ and random tape $F(\vec{J})$. For any choice of $1 \leq \sigma_i \leq N/k_i$ such that $\sigma_i k_i \in \mathbb{N}$ for $i \in [\mu]$, we have that

$$\begin{aligned}
\Pr\left[S_m\right] &= \sum_f \Pr\left[F = f\right] \Pr\left[S_m \;\middle|\; F = f\right] \\
&\geq \sum_f \Pr\left[F = f\right] \left(\Pr\left[S_{m+1} \;\middle|\; F = f\right] - (Q+1)\kappa_m(\sigma_m)\right) \\
&= \Pr\left[S_{m+1}\right] - (Q+1)\kappa_m(\sigma_m) \\
&\geq \varepsilon(\mathcal{P}^*) - (Q+1) \sum_{i=\mu+1}^{m} \kappa_i(\sigma_i).
\end{aligned}$$

The first inequality follows by Lemma H.1 and Lemma H.3, the second inequality by the induction hypothesis.

The argument for the expected number of invocations of $\mathcal{A}$ follows similarly as for the success probability, using Lemma H.2 and that we stop an invocation of $\mathcal{E}_{m+1}$ after a single invocation of $\mathcal{A}$ if the indices do not match. ◻

To simplify the presentation of our result in Theorem 4.1, we set $\sigma_i = 2$ for all $i \in [\mu]$. Then $\sigma k_i/((\sigma_i - 1)k_i + 1) \leq 2$.

H.3 Coordinate-Wise Extension

In this section, we generalize our extractor to the coordinate-wise setting. The extractor for coordinate-wise special-sound protocols in [FMN23] essentially runs the [AFK22] extractor for special sound protocols in each coordinate. We obtain our extractor by modifying their coordinate-wise extractor to potentially fail at the very end because of the predicates. The coordinate-wise abstract sampling game with predicates is presented in Game 4. As in Game 3, the predicates do not affect the expected number of entries sampled in the abstract game. We therefore only need to focus on analyzing the probability of winning the game.

For all $i \in [U], n \in [r]$ we define the functions

$$\begin{aligned}
a_i &: ([N]^r)^U \to \mathbb{N}_{\geq 0}, \\
&(\vec{j}_1, \ldots, \vec{j}_U) \mapsto \left| \left\{ \vec{j} \in [N]^r \;\middle|\; M(\vec{j}_1, \ldots, \vec{j}_{i-1}, \vec{j}, \vec{j}_{i+1}, \ldots, \vec{j}_U) = (1, i, \cdot) \right\} \right|, \\
a_{i,n} &: ([N]^r)^U \to \mathbb{N}_{\geq 0}, \\
&(\vec{j}_1, \ldots, \vec{j}_U) \mapsto \left| \left\{ j \in [N] \;\middle|\; \begin{array}{l} \vec{j} = (j_{i,1}, \ldots, j_{i,n-1}, j, j_{i,n+1}, \ldots, j_{i,r}), \\ M(\vec{j}_1, \ldots, \vec{j}_{i-1}, \vec{j}, \vec{j}_{i+1}, \ldots, \vec{j}_U) = (1, i, \cdot) \end{array} \right\} \right|.
\end{aligned}$$

Game 4: The coordinate-wise abstract sampling game

Parameters:

- $k, r, N, U \in \mathbb{N}$ and a set T.
- rU-dimensional array M with entries $M(\vec{j}_1, \ldots, \vec{j}_U) \in \{0,1\} \times [U] \times T$ for all $(\vec{j}_1, \ldots, \vec{j}_U) \in ([N]^r)^U$.
- Functions $f^{\mathsf{chal}}_{\ell,n} : ([N]^r)^{\ell-1} \times [N] \to \{0,1\}$ for all $\ell \in [k], n \in [r]$.
- Functions $(f^{\mathsf{prop}}, f^{\mathsf{bind}}_1, \ldots, f^{\mathsf{bind}}_r)$ with $f^{\mathsf{prop}} : T \times (T^r)^{k-2} \to \{0,1\}$ and $f^{\mathsf{bind}}_n : T \times (T^r)^{k-2} \times T \to \{0,1\}$ for all $n \in [r]$.

Game:

1. Sample $(\vec{j}_1, \ldots, \vec{j}_U) \xleftarrow{\$} ([N]^r)^U$ and compute $(v, i, t) = M(\vec{j}_1, \ldots, \vec{j}_U)$.
2. If $v = 0$, abort with output 0.
3. Set $\vec{s}_1 \coloneqq \vec{j}_i$, $t_1 \coloneqq t$. For each $n \in [r]$, set $\ell_n \coloneqq 1$ and $W_n \coloneqq [N] \setminus \{s_{1,n}\}$.
4. For each $n \in [r]$, repeat the following until $\ell_n = k$ or $W_n = \varnothing$:
   (a) Sample $j' \xleftarrow{\$} W_n$ and set $W_n \coloneqq W_n \setminus \{j'\}$.
   (b) Set $\vec{j}' \coloneqq (j_{i,1}, \ldots, j_{i,n-1}, j', j_{i,n+1}, \ldots, j_{i,r})$.
   (c) Compute $(v', i', t') = M(\vec{j}_1, \ldots, \vec{j}_{i-1}, \vec{j}', \vec{j}_{i+1}, \ldots, \vec{j}_U)$.
   (d) If $v' = 1$ and $i' = i$, set $\ell_n \coloneqq \ell_n + 1$, $s_{\ell_n,n} \coloneqq j'$ and $t_{\ell_n,n} \coloneqq t'$.
5. If $\ell_n < k$ for some $n \in [r]$, output 0.
6. For each $\ell \in [k] \setminus \{1\}$, set $\vec{s}_\ell \coloneqq (s_{\ell,1}, \ldots, s_{\ell,r})$ and $\vec{t}_\ell \coloneqq (t_{\ell,1}, \ldots, t_{\ell,r})$.
7. If $f^{\mathsf{chal}}_{\ell,n}((\vec{s}_1, \ldots \vec{s}_{\ell-1}), s_{\ell,n}) = 0$ for some $\ell \in [k], n \in [r]$, output 0.
8. If $f^{\mathsf{prop}}\left(t_1, \left(\vec{t}_2, \ldots, \vec{t}_{k-1}\right)\right) = (0, y)$ for some y and $\bigvee_{n=1}^r f^{\mathsf{bind}}_n\left(t_1, \left(\vec{t}_2, \ldots, \vec{t}_{k-1}\right), t_{k,n}\right) = 1$, output 0. Else, output 1.

$a_i(\vec{j}_1, \ldots, \vec{j}_U)$ counts the number of entries with $v = 1$ and index i in the r-dimensional array $M(\vec{j}_1, \ldots, \vec{j}_{i-1}, \cdot, \vec{j}_{i+1}, \ldots, \vec{j}_U)$, while $a_{i,n}(\vec{j}_1, \ldots, \vec{j}_U)$ counts the number of such entries in the 1-dimensional subarray where all coordinates except the n-th are fixed. Hence, $a_{i,n}(\vec{j}_1, \ldots, \vec{j}_U) \le a_i(\vec{j}_1, \ldots, \vec{j}_U)$ for all $n \in [r]$.

Lemma H.6. *Consider the game in Game 4. Let $\vec{J} = (\vec{J}_1, \ldots, \vec{J}_U)$ be uniformly distributed in $([N]^r)^U$, indicating the first entry sampled, and let $(V, I, T) = M(\vec{J})$. For each $i \in [U]$ and $n \in [r]$, let $A_i = a_i(\vec{J})$ and $A_{i,n} = a_{i,n}(\vec{J})$. Let $F^{\mathsf{chal}}_{\ell,n}$ be the random variable indicating the outcome of $f^{\mathsf{chal}}_{\ell,n}$. Let (F^{prop}, Y) denote the outcome of f^{prop} and F^{bind}_n the outcome of f^{bind}_n. Let $d^{\mathsf{chal}}_1, \ldots, d^{\mathsf{chal}}_k, d^{\mathsf{com}} \in [0,1]$ be numbers such that for all $i \in [U], n \in [r]$ and all $a \in \mathbb{N}$, $k \le a \le N$,*

$$\frac{d^{\mathsf{chal}}_\ell N}{a - \ell + 1} \ge \Pr\left[F^{\mathsf{chal}}_{\ell,n} = 0 \;\middle|\; \begin{matrix} V = 1, I = i, A_{i,n} = a, \bigcap_{n' \ne n} A_{i,n'} \ge k, \\ \bigcap_{\ell'=1}^{\ell-1} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell',n'} = 1 \end{matrix}\right] \quad \textit{and}$$

$$\frac{d^{\mathsf{com}} N}{a - k + 1} \ge \Pr\left[F^{\mathsf{bind}}_n = 1 \;\middle|\; \begin{matrix} V = 1, I = i, A_{i,n} = a, \bigcap_{n' \ne n} A_{i,n'} \ge k, \\ \bigcap_{\ell=1}^{k} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell,n'} = 1, F^{\mathsf{prop}} = 0, Y = n \end{matrix}\right].$$

Finally, set some tuning parameter $1 \le \sigma \le N/k$ such that $\sigma k \in \mathbb{N}$. Then the game ouputs 1 with probability at least

$$\Pr\left[V = 1\right] - rA \cdot \max\left(\frac{\sigma k - 1}{N}, \frac{\sigma k}{\sigma k - k + 1} d^{\mathsf{com}} + \sum_{\ell=1}^{k} \frac{\sigma k}{\sigma k - \ell + 1} d^{\mathsf{chal}}_\ell\right),$$

where $A = \sum_{i=1}^U \Pr\left[A_i > 0\right]$.

Proof. For the sake of conciseness, we let E_i denote the event that $V = 1, I = i$. By the description of the abstract game, we have that the probability that the game outputs 1 is

$$\sum_{i=1}^{U} \Pr\left[E_i, \bigcap_{n=1}^{r} A_{i,n} \geq k, \bigcap_{\ell=1}^{k} \bigcap_{n=1}^{r} F^{\mathsf{chal}}_{\ell,n} = 1, \left(F^{\mathsf{prop}} = 1 \cup \bigcup_{n=1}^{r} F^{\mathsf{bind}}_{n} = 0\right)\right]$$

This probability can be decomposed as follows.

$$= \Pr\left[V = 1\right] - \sum_{i=1}^{U} \Pr\left[E_i, \bigcup_{n=1}^{r} A_{i,n} < \sigma k\right] \tag{12}$$

$$- \sum_{i=1}^{U} \Pr\left[E_i, \bigcap_{n=1}^{r} A_{i,n} \geq \sigma k, \bigcup_{\ell=1}^{k} \bigcup_{n=1}^{r} F^{\mathsf{chal}}_{\ell,n} = 0\right] \tag{13}$$

$$- \sum_{i=1}^{U} \Pr\left[E_i, \bigcap_{n=1}^{r} A_{i,n} \geq \sigma k, \bigcap_{\ell=1}^{k} \bigcap_{n=1}^{r} F^{\mathsf{chal}}_{\ell,n} = 1, f^{\mathsf{prop}} = 0, \bigcap_{n=1}^{r} f^{\mathsf{bind}}_{n} = 1\right] \tag{14}$$

We upper bound each of the three sums individually. For the first sum (12), after applying a union bound, the argument is the same for each $A_{i,n}$.

$$\begin{aligned}
\sum_{i=1}^{U} \Pr\left[E_i, \bigcup_{n=1}^{r} A_{i,n} < \sigma k\right] &\leq \sum_{n=1}^{r} \sum_{i=1}^{U} \Pr\left[E_i, A_{i,n} < \sigma k\right] \\
&= \sum_{n=1}^{r} \sum_{i=1}^{U} \sum_{a=0}^{\sigma k - 1} \Pr\left[A_{i,n} = a\right] \frac{a}{N} \\
&\leq \sum_{n=1}^{r} \sum_{i=1}^{U} \sum_{a=1}^{\sigma k - 1} \Pr\left[A_{i,n} = a\right] \frac{\sigma k - 1}{N}
\end{aligned}$$

Next we bound the second sum (13).

$$\begin{aligned}
&\sum_{i=1}^{U} \Pr\left[E_i, \bigcap_{n=1}^{r} A_{i,n} \geq \sigma k, \bigcup_{\ell=1}^{k} \bigcup_{n=1}^{r} F^{\mathsf{chal}}_{\ell,n} = 0\right] \\
&= \sum_{i=1}^{U} \Pr\left[E_i, \bigcap_{n=1}^{r} A_{i,n} \geq \sigma k, \bigcup_{\ell=1}^{k} \bigcup_{n=1}^{r} \left(F^{\mathsf{chal}}_{\ell,n} = 0, \bigcap_{\ell'=1}^{\ell-1} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell',n'} = 1\right)\right] \\
&\leq \sum_{\ell=1}^{k} \sum_{n=1}^{r} \sum_{i=1}^{U} \Pr\left[E_i, \bigcap_{n'=1}^{r} A_{i,n'} \geq \sigma k, \bigcap_{\ell'=1}^{\ell-1} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell',n'} = 1, F^{\mathsf{chal}}_{\ell,n} = 0\right] \\
&\leq \sum_{\ell=1}^{k} \sum_{n=1}^{r} \sum_{i=1}^{U} \Pr\left[E_i, A_{i,n} \geq \sigma k, \bigcap_{n' \neq n} A_{i,n'} \geq k, \bigcap_{\ell'=1}^{\ell-1} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell',n'} = 1, F^{\mathsf{chal}}_{\ell,n} = 0\right] \\
&\leq \sum_{n=1}^{r} \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_{i,n} = a\right] \cdot \Pr\left[E_i \;\middle|\; A_{i,n} = a\right] \\
&\quad \cdot \sum_{\ell=1}^{k} \Pr\left[\bigcap_{n' \neq n} A_{i,n'} \geq k, \bigcap_{\ell'=1}^{\ell-1} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell',n'} = 1 \;\middle|\; E_i, A_{i,n} = a\right] \\
&\quad \cdot \Pr\left[F^{\mathsf{chal}}_{\ell,n} = 0 \;\middle|\; E_i, A_{i,n} = a, \bigcap_{n' \neq n} A_{i,n'} \geq k, \bigcap_{\ell'=1}^{\ell-1} \bigcap_{n'=1}^{r} F^{\mathsf{chal}}_{\ell',n'} = 1\right] \\
&\leq \sum_{n=1}^{r} \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_{i,n} = a\right] \cdot \frac{a}{N} \sum_{\ell=1}^{k} \frac{d^{\mathsf{chal}}_{\ell} N}{a - \ell + 1} \\
&\leq \sum_{n=1}^{r} \sum_{i=1}^{U} \sum_{a=\sigma k}^{N} \Pr\left[A_{i,n} = a\right] \sum_{\ell=1}^{k} \frac{\sigma k}{\sigma k - \ell + 1} d^{\mathsf{chal}}_{\ell}
\end{aligned}$$

Finally, we bound the third sum (14).

$$
\begin{aligned}
&\sum_{y=1}^{r}\sum_{i=1}^{U}\Pr\left[E_i,\bigcap_{n=1}^{r}A_{i,n}\geq\sigma k,\bigcap_{\ell=1}^{k}\bigcap_{n=1}^{r}F^{\mathsf{chal}}_{\ell,n}=1,f^{\mathsf{prop}}=0,Y=y,\bigcap_{n=1}^{r}f^{\mathsf{bind}}_{n}=1\right]\\
&\leq\sum_{y=1}^{r}\sum_{i=1}^{U}\sum_{a=\sigma k}^{N}\Pr\left[A_{i,y}=a\right]\cdot\Pr\left[E_i\mid A_{i,y}=a\right]\\
&\cdot\Pr\left[F^{\mathsf{bind}}_{y}=1\;\middle|\;E_i,A_{i,y}=a,\bigcap_{n\neq y}A_{i,n}\geq k,\bigcap_{\ell=1}^{k}\bigcap_{n=1}^{r}F^{\mathsf{chal}}_{\ell,n}=1,F^{\mathsf{prop}}=0,Y=y\right]\\
&\leq\sum_{y=1}^{r}\sum_{i=1}^{U}\sum_{a=\sigma k}^{N}\Pr\left[A_{i,y}=a\right]\frac{a}{N}\cdot\frac{d^{\mathsf{com}}N}{a-k+1}\\
&\leq\sum_{y=1}^{r}\sum_{i=1}^{U}\sum_{a=\sigma k}^{N}\Pr\left[A_{i,y}=a\right]\frac{\sigma k}{\sigma k-k+1}d^{\mathsf{com}}\\
&=\sum_{n=1}^{r}\sum_{i=1}^{U}\sum_{a=\sigma k}^{N}\Pr\left[A_{i,n}=a\right]\frac{\sigma k}{\sigma k-k+1}d^{\mathsf{com}}
\end{aligned}
$$

Combining the bounds for the three sums (12, 13, 14) and using that $\Pr\left[A_{i,n}>0\right]\leq\Pr\left[A_i>0\right]$ for all $n\in[r]$, the lemma follows. □

Lemma H.7 (Expected cost [FMN23, Lemma 8.1]). *Consider the game in Game 4, as well as a cost function $\Gamma:[N]^U\to\mathbb{R}_{\geq 0}$ and a constant cost $\gamma\in\mathbb{R}_{\geq 0}$. Let $\vec{J}=(\vec{J}_1,\ldots\vec{J}_U)$ be uniformly distributed in $([N]^r)^U$, indicating the first entry sampled, and let $(V,I,T)=M(\vec{J})$. For each $i\in[U]$, let $A_i=a_i(\vec{J})$. We define the cost of sampling an entry $M(j_1,\ldots,j_U)=(v,i,t)$ with index $i=I$ to be $\Gamma(j_1,\ldots,j_U)$ and the cost of sampling an entry $M(j_1,\ldots,j_U)=(v,i,t)$ with $i\neq I$ to be γ. Let Δ be the total cost of playing this game. Then*

$$\mathbb{E}[\Delta]\leq(1+r(k-1))\cdot\mathbb{E}[\Gamma(\vec{J})]+r(k-1)\cdot A'\cdot\gamma,$$

where $A'=\sum_{i=1}^{U}\Pr\left[I\neq i,A_i>0\right]\leq A$.

From this abstract game, an extractor for coordinate-wise predicate-special-sound protocols can be constructed in the same way as in Section H.2. We let $N=|S|$ be the cardinality of the base challenge space. $f^{\mathsf{chal}}_{\ell,n}$ is instantiated as the ℓ-th challenge predicate evaluated in the n-th coordinate. Note that there is a slight mismatch in their inputs. In the framework, the ℓ-th challenge predicate gets the $\ell-1$ first challenge vectors for every coordinate, $(\vec{s}_1,\vec{c}_2,\ldots,\vec{c}_{\ell-1})$. However, in the abstract game, $f^{\mathsf{chal}}_{\ell,n}$ only gets $\vec{s}_1$ and $s_{\ell',n,n}$ for every $\ell'\in[\ell]\setminus\{1\},n\in[r]$, it does not get the rest of $\vec{s}_{\ell',n}$. But since all the other entries of $\vec{s}_{\ell',n}$ have to be equal to those in $\vec{s}_1$, we do not need to pass them to the challenge predicate.

f^{bind}_{n} is the binding predicate evaluated in the n-th coordinate. Using Definition G.6, we let the $y\in[r]$ output by f^{prop} indicate the coordinate where we can use the failure density. Thus, Definition G.5 and Definition G.6 imply that we can set $d^{\mathsf{chal}}_{\ell}=p^{\mathsf{chal}}_{m,\ell}$ for all $\ell\in[k_m]$ and $d^{\mathsf{com}}_{m}=p^{\mathsf{com}}_{m}$. In addition, we can once again bound $A=\sum_{i=1}^{U}\Pr\left[A_i>0\right]\leq(Q+1)$ using Lemma H.3. In conclusion, by using Lemma H.6 and Lemma H.7, we obtain the following.

Proposition H.1. *Let $\Pi=(\mathcal{P},\mathcal{V})$ be a $(\boldsymbol{R},\boldsymbol{K},\boldsymbol{\Phi})$-coordinate-wise predicate-special-sound argument of knowledge for a relation R_{pp}. In addition, let $R_{\mathsf{bind,pp}}$ be a coordinate-wise binding relation for $\boldsymbol{\Phi}$. Consider the adaptive Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of Π. There exists a knowledge extractor for the relation $R_{\mathsf{pp}}\cup R_{\mathsf{bind,pp}}$, which given black-box access to an adaptive Q-query random oracle prover $\mathcal{P}^*$ for $\mathsf{FS}[\Pi]$, succeeds with probability at least*

$$\varepsilon(\mathcal{P}^*)-(Q+1)\sum_{i=1}^{\mu}r_i\min_{\sigma_i\in[1,\frac{N}{k_i}]:\ \sigma_i k_i\in\mathbb{N}}\kappa_i(\sigma_i),$$

where

$$\kappa_i(\sigma)=\max\left(\frac{\sigma k_i-1}{N},\frac{\sigma k_i}{\sigma k_i-k_i+1}p^{\mathsf{com}}_{i}+\sum_{\ell=1}^{k_i}\frac{\sigma k_i}{\sigma k_i-\ell+1}p^{\mathsf{chal}}_{i,\ell}\right).$$

The number of times that the knowledge extractor invokes $\mathcal{P}^*$ is in expectation at most $K_m+Q(K_m-1)$, where $K=\prod_{i=1}^{\mu}(r_i(k_i-1)+1)$

I Proof of Non-Interactive Knowledge Soundness of LaBRADOR

In this section, we provide the first *full* proof of non-interactive knowledge soundness of LaBRADOR [BS23]. In our soundness analysis of LaBRADOR, we initially focus on a single iteration. Due to the structure of PSS, the soundness of recursively composed protocols then follows easily. A single iteration of LaBRADOR is $(\boldsymbol{R}, \boldsymbol{K}, \boldsymbol{\Phi})$-coordinate-wise predicate-special-sound, with $\boldsymbol{R} = (1, 1, 1, r)$, $\boldsymbol{K} = (2, 2, 2, 3)$ and r the number of witnesses. We now define the predicate system $\boldsymbol{\Phi}$ from the bottom up, for the sake of simplicity we treat rounds which only have one coordinate as if they were simply predicate-special-sound.

We are interested in the case where LaBRADOR is instanted with a ring $\mathcal{R}_q$ of degree d with splitting factor l and a B-well-spread challenge set, where each challenge has operator norm at most T_{op}. Our proof may be easily adapted for a ring that permits a challenge space where challenge differences are always invertible, as it was the case for the original LaBRADOR [BS23]. For our statements we require an ℓ_2-norm bound $\sqrt{\lambda/C_2}\beta$ at most q/C_1, where C_1, C_2 are the parameters of Lemma 2.2.

I.1 Analysing Levels

Recall that LaBRADOR is a $(2\mu + 1)$-message public-key argument of knowledge with $\mu = 4$. Working bottom up, we define predicates for one layer at a time. To analyze the failure density of the predicates on some level, we may by definition assume that the predicates on the levels below it are satisfied. Thus, we incrementally enforce stronger properties on the extracted candidate witness.

We begin with a single transcript $t_5 \in \mathbb{T}_5$. As always, if t_5 is an accepting transcript then $\boldsymbol{\Phi}_5(t_5) = 1$. For all remaining levels $(i = 1, 2, 3, 4)$, we set $\Phi_i^{\mathsf{com}} = (\Phi_i^{\mathsf{prop}}, \Phi_i^{\mathsf{bind}})$.

I.1.1 Level 4.

This round contains the amortized opening to the commitment, and is the only level which makes use of the coordinate-wise extension of PSS and non-trivial challenge predicates. In particular, we have one coordinate for each witness vector we wish to extract, giving r coordinates in total. Let $(t_1, \vec{t}_2, \vec{t}_3) \in \mathbb{T}_4$ be the extracted tree with $\vec{t}_2 = (t_{2,1}, \dots, t_{2,r}) \in \mathbb{T}_5^r$ and $\vec{t}_3 = (t_{3,1}, \dots, t_{3,r}) \in \mathbb{T}_5^r$, such that $t_1, t_{2,i}, t_{3,i}$ are the transcripts for the ith coordinate. We let $\boldsymbol{c}_{1,1}, \dots, \boldsymbol{c}_{1,r}$ be the level 4 challenges of t_1, and let $\boldsymbol{c}_{i,j}$ be the j-th challenge of $t_{i,j}$ for $i \in \{2, 3\}, j \in [r]$. Note that the other challenges of the $t_{i,j}$ are the same as those in t_1.

To later enforce that the $\boldsymbol{g}_{i,j}$ garbage polynomials are computed correctly, our trick is to enforce that all the first challenges are units. For $i \in [r]$,

$$\Phi_{4,1}^{\mathsf{chal}}(i, \boldsymbol{c}_{1,i}) = 1 \;\Leftrightarrow\; \boldsymbol{c}_{1,i} \in \mathcal{R}_q^{\times}.$$

We also require that the difference between the first two challenges in each coordinate is invertible, so that we can compute weak openings for the inner commitments. For $i \in [r]$,

$$\Phi_{4,2}^{\mathsf{chal}}((\boldsymbol{c}_{1,1}, \dots, \boldsymbol{c}_{1,r}), i, \boldsymbol{c}_{2,i}) = 1 \;\Leftrightarrow\; (\boldsymbol{c}_{1,i} - \boldsymbol{c}_{2,i}) \in \mathcal{R}_q^{\times}.$$

Next, we define the commitment predicate to ensure that each garbage polynomial is computed correctly. We introduce some definition to describe the elements in each transcript. For $t \in \mathbb{T}_5$, we let $\vec{\boldsymbol{z}}(t)$, $\vec{\boldsymbol{v}}_i(t)$, $\boldsymbol{g}_{i,j}(t)$, $\boldsymbol{h}_{i,j}(t)$ and $\vec{\boldsymbol{\varphi}}_i(t)$ denote the corresponding values in the transcript t. Let

$$\bar{\boldsymbol{c}}_i = \boldsymbol{c}_{1,i} - \boldsymbol{c}_{2,i} \text{ and } \vec{\boldsymbol{w}}_i^* = \bar{\boldsymbol{c}}_i^{-1}\left(\vec{\boldsymbol{z}}(t_1) - \vec{\boldsymbol{z}}(t_{2,i})\right) \;\forall i \in [r].$$

Recall that by the LaBRADOR verification algorithm Protocol 3, $\|\vec{\boldsymbol{z}}(t)\|_2 \le (b+1)\beta'$. When the inner commitments are the same in t_1 and $t_{2,i}$, $\bar{\boldsymbol{c}}_i$ and $\vec{\boldsymbol{w}}_i^*$ form a weak opening for $\vec{\boldsymbol{v}}_i$ of norm $2(b+1)\beta'$. Furthermore, for $i, j \in [r]$, we need the prover to be bound to

$$\begin{aligned} \vec{\boldsymbol{y}}_i(t) &= \vec{\boldsymbol{z}}(t) - \mathsf{chal}_{4,i}(t)\vec{\boldsymbol{w}}_i^*, \\ \vec{\boldsymbol{y}}_{i,j}(t) &= \vec{\boldsymbol{z}}(t) - \mathsf{chal}_{4,i}(t)\vec{\boldsymbol{w}}_i^* - \mathsf{chal}_{4,j}(t)\vec{\boldsymbol{w}}_j^*, \end{aligned}$$

where $\mathsf{chal}_{4,i}(t)$ outputs the ith coordinate challenge for the fourth level. Finally, we define an extraction algorithm for each coordinate $\ell \in [r]$,

$$E_5(t, \ell) = \left(\left(\vec{\boldsymbol{v}}_i(t), \boldsymbol{g}_{i,j}(t), \boldsymbol{h}_{i,j}(t)\right)_{i,j \in [r]}, \vec{\boldsymbol{y}}_\ell(t), \left(\vec{\boldsymbol{y}}_{\ell,j}(t)\right)_{j \in [r], \ell \neq j} \right).$$

The property predicate ensures that the garbage polynomials are computed correctly with respect to the $\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*$ computed from t_1 and $\vec{t}_2$.

$$\Phi_4^{\mathsf{prop}}(t_1, \vec{t}_2) = 1 \;\Leftrightarrow\; \forall i, j \in [r] :\; \boldsymbol{h}_{i,i}(t_1) = \langle \vec{\boldsymbol{\varphi}}_i(t_1), \vec{\boldsymbol{w}}_i^* \rangle \;\wedge\; \boldsymbol{g}_{i,j}(t_1) = \langle \vec{\boldsymbol{w}}_i^*, \vec{\boldsymbol{w}}_j^* \rangle.$$

The binding predicate states that the relevant openings should be consistent across transcripts.

$$\Phi_4^{\mathsf{bind}}((t_1, \vec{t}_2), \ell, t_{3,\ell}) = 1 \Leftrightarrow E_5(t_1, \ell) = E_5(t_{2,\ell}, \ell) = E_5(t_{3,\ell}, \ell).$$

In preparation for the next level, for $t_4 = (t_1, \vec{t}_2, \vec{t}_3)$ we let $E_4(t_4) = (\vec{\boldsymbol{w}}_i^*, \bar{\boldsymbol{c}}_i)_{i\in[r]}$.

Lemma I.1. *In the fourth level $\Phi_{4,1}^{\mathsf{chal}}$ has failure density $p_{4,1}^{\mathsf{chal}} = lB$, $\Phi_{4,2}^{\mathsf{chal}}$ has failure density $p_{4,2}^{\mathsf{chal}} = lB$ and Φ_4^{com} has failure density $p_4^{\mathsf{com}} = 5B$.*

Proof. We address each of the predicates individually, starting with $\Phi_{4,1}^{\mathsf{chal}}$. Given a fixed $\boldsymbol{c}^* \in \mathcal{R}_q$, Lemma 3.1 tells us that

$$\Pr_{\boldsymbol{c} \xleftarrow{\$} \mathcal{C}} \left[\boldsymbol{c}^* - \boldsymbol{c} \notin \mathcal{R}_q^\times \right] \le lB.$$

Letting $\boldsymbol{c}^* = \mathbf{0}$, it follows that a uniformly sampled challenge in $\mathcal{C}$ is a unit with probability less than lB. Hence, there are at most $lB|\mathcal{C}|$ non-units in $\mathcal{C}$, meaning that the failure density is $p_{4,1}^{\mathsf{chal}} = lB$.

A similar argument may be applied for $\Phi_{4,2}^{\mathsf{chal}}$ in each coordinate. Let $\boldsymbol{c}^* = \boldsymbol{c}_{1,i}$, then the lemma implies that at most $lB|\mathcal{C}|$ of the challenges $\boldsymbol{c} \in \mathcal{C}$ are such that $\boldsymbol{c}_{1,i} - \boldsymbol{c} \notin \mathcal{R}_q^\times$. One of these is $\boldsymbol{c}_{1,i}$ itself, which we can exclude as it will not be sampled again by the extractor. Thus, we obtain the failure density $lB - 1/|\mathcal{C}|$. However, for simplicity, we upper bound the failure density of $\Phi_{4,2}^{\mathsf{chal}}$ by $p_{4,2}^{\mathsf{chal}} = lB$.

We now proceed to Φ_4^{com}. To derive the failure density of this predicate, we analyze the $\boldsymbol{h}_{i,i}$ and $\boldsymbol{g}_{i,j}$ separately. By a union bound, the failure density of this predicate is upper bounded by the sum of the failure density of each case.

Assume for some index $i_0 \in [r]$, $\boldsymbol{h}_{i_0,i_0}(t_1) \neq \langle \vec{\boldsymbol{\varphi}}_{i_0}(t_1), \vec{\boldsymbol{w}}_{i_0}^* \rangle$. Then we upper bound the number of choices of the challenge $\boldsymbol{c}_{i_0}$ in the i_0th coordinate which could lead to an accepting transcript without breaking binding. To do this, we are going to use the Schwartz-Zippel lemma (Lemma 3.2). We are going to redefine the verification equation for the $\boldsymbol{h}_{i,j}$ as a non-zero polynomial which has $\boldsymbol{c}_{i_0}$ as a root. The verification algorithm for a $t \in \mathbb{T}_5$ has the following check for the $\boldsymbol{h}_{i,j}$.

$$\sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i(t), \vec{\boldsymbol{z}}(t) \rangle \boldsymbol{c}_i \stackrel{?}{=} \sum_{i,j=1}^{r} \boldsymbol{h}_{i,j}(t) \boldsymbol{c}_i \boldsymbol{c}_j,$$

with $\boldsymbol{c}_i = \mathsf{chal}_{4,i}(t)$. The polynomial for the Schwartz-Zippel test is obtained by rearranging this equation, substituting $\vec{\boldsymbol{z}}(t)$ with $\vec{\boldsymbol{y}}_{i_0}(t) + \boldsymbol{c}_{i_0} \vec{\boldsymbol{w}}_{i_0}^*$ and setting the i_0-th challenge to be the variable X_{i_0}.

$$\begin{aligned}
&\left(\sum_{\substack{1 \le i,j \le r \\ i,j \neq i_0}} \boldsymbol{h}_{i,j}(t) \boldsymbol{c}_i \boldsymbol{c}_j - \sum_{\substack{1 \le i \le r \\ i \neq i_0}} \langle \vec{\boldsymbol{\varphi}}_i(t), \vec{\boldsymbol{y}}_{i_0}(t) \rangle \boldsymbol{c}_i \right) \\
&+ \left(\sum_{\substack{1 \le i \le r \\ i \neq i_0}} (\boldsymbol{h}_{i,i_0}(t) + \boldsymbol{h}_{i_0,i}(t)) \boldsymbol{c}_i - \langle \vec{\boldsymbol{\varphi}}_{i_0}(t), \vec{\boldsymbol{y}}_{i_0}(t) \rangle - \sum_{\substack{1 \le i \le r \\ i \neq i_0}} \langle \vec{\boldsymbol{\varphi}}_i(t), \vec{\boldsymbol{w}}_{i_0}^* \rangle \boldsymbol{c}_i \right) X_{i_0} \\
&+ \left(\boldsymbol{h}_{i_0,i_0}(t) - \langle \vec{\boldsymbol{\varphi}}_{i_0}(t), \vec{\boldsymbol{w}}_{i_0}^* \rangle \right) X_{i_0}^2 \stackrel{?}{=} \mathbf{0}.
\end{aligned}$$

The left hand side is a polynomial over $\mathcal{R}_q[X_{i_0}]$ of degree 2, as the leading coefficient by assumption is not $\mathbf{0}$. For t to be accepting, the i_0-th challenge $\boldsymbol{c}_{i_0}$ must be a root of this polynomial. Given $(t_1, \vec{t}_2)$, we have fixed all the values in the coefficients of this polynomial. The challenges $\boldsymbol{c}_j$ for $j \neq i_0$ are fixed by the coordinate-wise extraction. The elements $\vec{\boldsymbol{w}}_1^*, \ldots, \vec{\boldsymbol{w}}_r^*$ are just the constants we chose to define $\vec{\boldsymbol{y}}_i$. The other values are fixed by the binding predicate. Thus, if the third transcript t_{3,i_0} succeeds without breaking binding, it must do so using a $\boldsymbol{c}_{i_0}$ that is a root of this *fixed* polynomial.

By Lemma 3.2, the probability that a challenge that was sampled independently and uniformly at random from $\mathcal{C}$ is a root of a fixed polynomial of degree 2 is at most $2B$. It follows that there can be at most $2B \cdot |\mathcal{C}|$ roots of the polynomial. Thus, the failure density for this case is $2B$.

Next, assume $\boldsymbol{g}_{i_0,i_0} \neq \langle \vec{\boldsymbol{w}}_{i_0}^*, \vec{\boldsymbol{w}}_{i_0}^* \rangle$. Then the argument is exactly the same as before. The verification algorithm checks that the $\boldsymbol{g}_{i,j}$ in a transcript $t \in \mathbb{T}_5$ satisfy the equation

$$\langle \vec{\boldsymbol{z}}(t), \vec{\boldsymbol{z}}(t) \rangle \stackrel{?}{=} \sum_{i,j=1}^{r} \boldsymbol{g}_{i,j}(t) \boldsymbol{c}_i \boldsymbol{c}_j, \tag{15}$$

where $\boldsymbol{c}_i = \mathsf{chal}_{4,i}(t)$. This equation can be reformulated to a non-zero polynomial of degree 2 which must have $\boldsymbol{c}_{i_0}$ as a root in order for the transcript to be accepting. Thus, we get the failure density $2B$ for this case too.

Finally, assume $\boldsymbol{g}_{i_0,j_0}(t_1) \neq \langle \vec{\boldsymbol{w}}^*_{i_0}, \vec{\boldsymbol{w}}^*_{j_0} \rangle$ for a $j_0 \neq i_0$, but that all the $\boldsymbol{g}_{i,i}(t_1)$ were computed correctly. By reorderring (15), substituting $\vec{\boldsymbol{z}}(t)$ with $\vec{\boldsymbol{y}}_{i_0,j_0}(t) + \boldsymbol{c}_{i_0}\vec{\boldsymbol{w}}^*_{i_0} + \boldsymbol{c}_{j_0}\vec{\boldsymbol{w}}^*_{j_0}$ and setting $\boldsymbol{c}_{i_0}$ to be X_{i_0} and $\boldsymbol{c}_{j_0}$ to be X_{j_0}, we obtain the following bivariate polynomial.

$$
\begin{aligned}
&\left(\sum_{i,j \in \{i_0,j_0\}} \boldsymbol{g}_{i,j}(t)\boldsymbol{c}_i\boldsymbol{c}_j - \langle \vec{\boldsymbol{y}}_{i_0,j_0}(t), \vec{\boldsymbol{y}}_{i_0,j_0}(t) \rangle \right) \\
&+ 2 \left(\sum_{i \in \{i_0,j_0\}} \boldsymbol{g}_{i_0,i}(t)\boldsymbol{c}_i - \langle \vec{\boldsymbol{y}}_{i_0,j_0}(t), \vec{\boldsymbol{w}}^*_{i_0} \rangle \right) X_{i_0} \\
&+ 2 \left(\sum_{i \in \{i_0,j_0\}} \boldsymbol{g}_{j_0,i}(t)\boldsymbol{c}_i - \langle \vec{\boldsymbol{y}}_{i_0,j_0}(t), \vec{\boldsymbol{w}}^*_{j_0} \rangle \right) X_{j_0} \\
&+ 2 \left(\boldsymbol{g}_{i_0,j_0}(t) - \langle \vec{\boldsymbol{w}}^*_{i_0}, \vec{\boldsymbol{w}}^*_{j_0} \rangle \right) X_{i_0} X_{j_0} \\
&+ \left(\boldsymbol{g}_{i_0,i_0}(t) - \langle \vec{\boldsymbol{w}}^*_{i_0}, w^*_{i_0} \rangle \right) X^2_{i_0} + \left(\boldsymbol{g}_{j_0,j_0}(t) - \langle \vec{\boldsymbol{w}}^*_{j_0}, \vec{\boldsymbol{w}}^*_{j_0} \rangle \right) X^2_{j_0}.
\end{aligned}
$$

The coefficients of this polynomial are once again fixed by binding and coordinate-wise extraction.

The degree of the polynomial is 2, since $\boldsymbol{g}_{i_0,j_0}(t_1) - \langle \vec{\boldsymbol{w}}^*_{i_0}, \vec{\boldsymbol{w}}^*_{j_0} \rangle \neq 0$. However, when extracting in the i_0-th coordinate, we always use the same j_0-th coordinate $\boldsymbol{c}_{j_0}$ Thus, we do not evaluate the polynomial in a j_0-th challenge chosen independently of the polynomial, which is a requirement to use the Schwartz-Zippel lemma.

We circumvent this by considering the polynomial with X_{j_0} evaluated in $\boldsymbol{c}_{j_0}$. Recall, by $\varPhi^{\mathsf{chal}}_{4,1}$ we know that the challenge in each coordinate is a unit $\boldsymbol{c}_{j_0} \in \mathcal{R}^\times_q$. As a result $\left(\boldsymbol{g}_{i_0,j_0}(t) - \langle \vec{\boldsymbol{w}}^*_{i_0}, \vec{\boldsymbol{w}}^*_{j_0} \rangle \right) \boldsymbol{c}_{j_0} \neq 0$, meaning that the remaining polynomial is not the zero polynomial. This allows us to consider a polynomial in $\mathcal{R}_q[X_{i_0}]$ of degree 1 with coefficients independent of $\boldsymbol{c}_{i_0}$. By the Schwartz-Zippel lemma, it follows that the failure density in this case is B. Summing over all cases, we obtain $p^{\mathsf{com}}_4 = 5B$. □

I.1.2 Level 3. Now we proceed to the third level. Here individual constraints are aggregated to form one single function F. It is computed as the random linear combination of the K constraints in $\mathcal{F}$ and the $K'' = \lceil \lambda / \log q \rceil$ aggregated constant term dot product constraints.

$$
\begin{aligned}
F(\vec{\boldsymbol{w}}^*_1, \ldots, \vec{\boldsymbol{w}}^*_r) &= \sum_{k=1}^{K} \boldsymbol{\alpha}_k f^{(k)}(\vec{\boldsymbol{w}}^*_1, \ldots, \vec{\boldsymbol{w}}^*_r) + \sum_{k=1}^{K''} \boldsymbol{\beta}_k f''^{(k)}(\vec{\boldsymbol{w}}^*_1, \ldots, \vec{\boldsymbol{w}}^*_r) \\
&= \sum_{i,j}^{r} \boldsymbol{a}_{i,j} \langle \vec{\boldsymbol{w}}^*_i, \vec{\boldsymbol{w}}^*_j \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}^*_i \rangle - \boldsymbol{b}.
\end{aligned}
$$

The verification algorithm has the following check.

$$
\sum_{i,j=1}^{r} \boldsymbol{a}_{i,j}\boldsymbol{g}_{i,j} + \sum_{i=1}^{r} \boldsymbol{h}_{i,i} - \boldsymbol{b} \stackrel{?}{=} \boldsymbol{0}. \tag{16}
$$

We wish to ensure that all of the aggregated functions evaluate to $\boldsymbol{0}$. Given $t \in \mathbb{T}_4$ where $\boldsymbol{\varPhi}(t) = 1$, we know E_4 will extract weak openings for the inner commitments.

$$
\varPhi^{\mathsf{prop}}_3(t_1) = 1 \;\Leftrightarrow\; \begin{cases} (\vec{\boldsymbol{w}}^*_i, \bar{\boldsymbol{c}}_i)_{i \in [r]} \leftarrow E_4(t_1), \\ \forall f \in \mathcal{F}: \; f(\vec{\boldsymbol{w}}^*_1, \ldots, \vec{\boldsymbol{w}}^*_r) = \boldsymbol{0}, \\ \forall f'' \in \mathcal{F}'': \; f''(\vec{\boldsymbol{w}}^*_1, \ldots, \vec{\boldsymbol{w}}^*_r) = \boldsymbol{0}. \end{cases}
$$

Binding holds when the extracted witnesses and inner commitments are the same in each subtree. We make the same requirements for rounds $\ell = 1, 2, 3$:

$$
\varPhi^{\mathsf{bind}}_\ell(t^1_{\ell+1}, t^2_{\ell+1}) = 1 \Leftrightarrow \begin{cases} (\vec{\boldsymbol{w}}^*_{i,k}, \bar{\boldsymbol{c}}_{i,k})_{i \in [r]} \leftarrow E_{\ell+1}(t^k_{\ell+1}) \text{ for } k \in \{1,2\} \\ \forall i \in [r], \vec{\boldsymbol{w}}^*_{i,1} = \vec{\boldsymbol{w}}^*_{i,2}, \; \vec{\boldsymbol{v}}_{i,1} = \vec{\boldsymbol{v}}_{i,2} \end{cases}
$$

Here $\vec{\boldsymbol{v}}_{i,1}, \vec{\boldsymbol{v}}_{i,2}$ denotes the ith inner commitment in respectively $t^1_{\ell+1}$ and $t^2_{\ell+1}$. Note that our analysis assumes the predicates on the lower levels are satisfied, so that in particular, the prover does not violate binding with respect to the inner commitments in $t^1_{\ell+1}$ or $t^2_{\ell+1}$. Hence, there is a unique inner commitment in each subtree. where for $t_\ell = (t^1_{\ell+1}, t^2_{\ell+1})$ we let $E_\ell(t_\ell) = E_{\ell+1}(t^1_{\ell+1})$.

Lemma I.2. *The failure density of Φ_3^{com} in $\boldsymbol{\Phi}$ is $p_3^{\mathsf{com}} = q^{-d/l}$.*

Proof. Assume Φ_3^{prop} does not hold. The analysis of the functions in $\mathcal{F}$ and $\mathcal{F}''$ is the same, so without loss of generality, we focus on the former. Then there is some $f^{(k_0)} \in \mathcal{F}$ such that $f^{(k_0)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) \neq \mathbf{0}$.

Let us bound the fraction of challenges that can be used to produce a second valid tree without breaking binding. The evaluation of $f^{(k_0)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)$ is fixed by the binding predicate Φ_4^{bind}. Let

$$F(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) = \boldsymbol{\alpha}_{k_0} f^{(k_0)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}^*) + F'(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*),$$

where F' is the sum of the other terms. Due to Φ_4^{com} we know that if $\boldsymbol{\Phi}_4(t_2) = 1$, we must have $\boldsymbol{g}_{i,j} = \langle \vec{\boldsymbol{w}}_i^*, \vec{\boldsymbol{w}}_j^* \rangle$ and $\boldsymbol{h}_{i,i} = \langle \vec{\boldsymbol{\varphi}}_i, \vec{\boldsymbol{w}}_i^* \rangle$. Thus, Equation 16 must hold for t_2, meaning $F(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) = \mathbf{0}$.

For F to evaluate to $\mathbf{0}$ over $\mathcal{R}_q$, it must evaluate to 0 in every CRT slot. $f^{(k_0)}(\vec{\boldsymbol{w}}^*) \neq \mathbf{0}$ over $\mathcal{R}_q$ means that there is some $i \in \{1, \dots, l\}$ such that its ith CRT slot is non-zero. Since the evaluation of $f^{(k_0)}$ is fixed by binding, this slot i is fixed.

Imagine that $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}$ were chosen independently and uniformly at random. The probability over the choice of $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}$ that F evaluates to $\mathbf{0}$ in $\mathcal{R}_q$ is upper bounded by the probability that it evaluates to $\mathbf{0}$ in the ith CRT slot. Each CRT slot is a field of order $q^{d/l}$. In a field, the product of an independent uniformly random element with a non-zero constant is still independently uniformly random. Hence, $\boldsymbol{\alpha}_{k_0} f^{(k_0)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)$ is an independently distributed uniformly random term of $F(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)$ in the ith CRT slot. This means that the evaluation of F in the ith CRT slot is uniformly random over the choice of $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}$. Thus, $F(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) = \mathbf{0} \bmod (X^{d/l} - \zeta_i)$ with probability $q^{-d/l}$. It follows that there are at most $q^{-d/l} \cdot |\mathbb{Z}_q^K \times \mathbb{Z}_q^{K''}|$ choices of $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}$ such that F evaluates to $\mathbf{0}$. We conclude that the failure density of this predicate is $q^{-d/l}$. ◻

I.1.3 Level 2.

On this level we wish to ensure that our extracted witnesses satisfy the constant term constraint functions in $\mathcal{F}'$ and that the projection was computed correctly.

$$\Phi_2^{\mathsf{prop}}(t_1) = 1 \;\Leftrightarrow\; \begin{cases} (\vec{\boldsymbol{w}}_i^*, \vec{\boldsymbol{c}}_i)_{i \in [r]} \leftarrow E_3(t_1), \\ \forall f' \in \mathcal{F}' : \; \mathsf{ct}(f'(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)) = 0 \bmod q, \\ \forall l \in [2\lambda] : \; p_l = \mathsf{ct}(\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(l)}), \vec{\boldsymbol{w}}_i^* \rangle) \bmod q \end{cases}.$$

Lemma I.3. *The failure density of Φ_2^{com} in $\boldsymbol{\Phi}$ is $p_2^{\mathsf{com}} = q^{-\lceil \lambda / \log q \rceil}$.*

Proof. In the first aggregation step, the constraints in $\mathcal{F}''$ are aggregated to $K'' = \lceil \lambda / \log_2 q \rceil$ functions by computing random linear combinations. Let $L = |\mathcal{F}'|$. For $k = 1, \dots, K''$,

$$\begin{aligned} f''^{(k)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) &= \sum_{l=1}^{L} \psi_l^{(k)} f'^{(l)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) + \sum_{l=1}^{2\lambda} \omega_l^{(k)} (\langle \sigma_{-1}(\vec{\boldsymbol{\pi}}_i^{(l)}), \vec{\boldsymbol{w}}_i^* \rangle - p_l) \\ &= \sum_{i,j=1}^{r} \boldsymbol{a}''^{(k)} \langle \vec{\boldsymbol{w}}_i^*, \vec{\boldsymbol{w}}_j^* \rangle + \sum_{i=1}^{r} \langle \vec{\boldsymbol{\varphi}}_i''^{(k)}, \vec{\boldsymbol{w}}_i^* \rangle - b_0''^{(k)}. \end{aligned}$$

Then the prover should extend $b_0''^{(k)}$ to $\boldsymbol{b}''^{(k)}$ so that $f''^{(k)}$ evaluates to $\mathbf{0}$ over $\mathcal{R}_q$ in the witness. The verification algorithm checks that every $b_0''^{(k)}$ is computed correctly. Because of Φ_3^{prop}, we know that $f''^{(k)}(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) = \mathbf{0}$ for all k, meaning that the rest of $\boldsymbol{b}''^{(k)}$ was computed correctly as well.

To compute the failure density of this predicate, assume without loss of generality that some $f'^{(l)} \in \mathcal{F}''$ does not evaluate to $\mathbf{0}$. The prover is bound to this evaluation by the binding predicate. Imagine that for the second subtree, $\vec{\psi}^{(k)} \xleftarrow{\$} \mathbb{Z}_q^L$ and $\vec{\omega}^{(k)} \xleftarrow{\$} \mathbb{Z}_q^{2\lambda}$. $\boldsymbol{b}''^{(k)}$ was computed honestly, so $b_0''^{(k)}$ contains the term $\psi_l^{(k)} f'(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)$, which is independently uniformly random since $\mathbb{Z}_q$ is a field. Thus, $b_0''^{(k)}$ is distributed uniformly at random and would only be the correct with probability $1/q$. The probability that all of the $\lceil \lambda / \log_2 q \rceil$ repetitions will have the correct value is then $q^{-\lceil \lambda / \log q \rceil}$. It follows that the proportion of choices of $\vec{\psi}^{(k)}$, $\vec{\omega}^{(k)}$ such that there is a valid subtree is $q^{-\lceil \lambda / \log q \rceil}$. ◻

I.1.4 Level 1. For the top level, we enforce that the witness is short.

$$\Phi_1^{\mathsf{prop}}(t_1) = 1 \;\Leftrightarrow\; (\vec{\boldsymbol{w}}_i^*, \bar{\boldsymbol{c}}_i)_{i\in[r]} \leftarrow E_2(t_1),\; \sum_{i=1}^r \left\| \vec{\boldsymbol{w}}_i^* \right\| \le \sqrt{\lambda/C_2}\beta.$$

Here C_2 is a constant defined as in Lemma 2.2.

We remark that to reduce the completeness error of LaBRADOR the first round of Fiat-Shamir transform should be slightly adjusted: (1) prover initializes $\mathsf{ctr} = 0$, (2) query H_1 with $(\mathsf{x}, \vec{\boldsymbol{u}}_1, \mathsf{ctr})$ (with x the statement to be proven) to obtain a random projection challenge, (3) compute $\vec{p}$, (4) if $\vec{p}$ exceeds the verification bound, set $\mathsf{ctr} = \mathsf{ctr} + 1$ and go to (2); else, proceed to the second round. The verifier must ensure $\mathsf{ctr} \in \{0, \dots, \lambda\}$. By viewing a tuple $(\vec{\boldsymbol{u}}_1, \mathsf{ctr})$ as a first-round message, the failure density for a single projection may be used without modification. For an honest prover the projection will be smaller than the required bound with probability 1/2. When this is not the case the prover will fail, giving completeness error 1/2. To address this we may either require the prover to start over, or alternatively, give the prover λ choices of projection, where the prover sends the first successful projection, resulting in negligible soundness error. By a union bound the failure density would then increase by a factor λ.

Lemma I.4. *The failure density of Φ_1^{com} in $\boldsymbol{\Phi}$ is $p_1^{\mathsf{com}} = 2^{-\lambda}$.*

Proof. Assume that $\sum_{i=1}^r \left\| \vec{\boldsymbol{w}}_i^* \right\|_2^2 > (\lambda/C_2)\beta^2$. By the previous predicates, we know that the prover computed their projection $\vec{p}$ honestly, so that $\vec{p} = \sum_{i=1}^r \Pi_i \vec{\boldsymbol{w}}_i^*$. Because the subtree is valid, it must be the case that $\|\vec{p}\|_2 \le \sqrt{\lambda}\beta$. The modular Johnson-Lindenstrauss lemma (Lemma 2.2) states the following. For a fixed vector in $\mathbb{Z}_q^m$ with ℓ_2-norm strictly greater than $\sqrt{\lambda/C_2}\beta$, the probability that an independently sampled projection challenge gives $\|\vec{p}\|_2 \le \sqrt{\lambda}\beta$ is at most $2^{-\lambda}$. Thus, the failure density of this predicate is $2^{-\lambda}$. □

I.2 PSS of LaBRADOR.

We modify the primary LaBRADOR relation to allow some norm slack σ,

$$R_\sigma = \left\{ \left(\begin{matrix} (\mathcal{F}, \mathcal{F}', \beta), \\ (\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) \end{matrix} \right) \middle| \begin{matrix} \forall f \in \mathcal{F},\; f(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*) = \mathbf{0}, \\ \forall f' \in \mathcal{F}',\; \mathsf{ct}(f'(\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)) = 0 \mod q, \\ \sum_{i=1}^r \left\| \vec{\boldsymbol{w}}_i^* \right\|_2^2 \le \sigma\beta^2 \end{matrix} \right\}. \tag{17}$$

Lemma I.5. *Let Π be the base LaBRADOR protocol as described in Protocol 2 and 3. Let $\boldsymbol{K} = (2, 2, 2, 3)$, $\boldsymbol{R} = (1, 1, 1, r)$, and let $\boldsymbol{\Phi}$ be the predicate system consisting of commitment predicates $\Phi_1^{\mathsf{com}}, \Phi_2^{\mathsf{com}}$, $\Phi_3^{\mathsf{com}}, \Phi_4^{\mathsf{com}}$ and the challenge predicates $\Phi_{4,1}^{\mathsf{chal}}, \Phi_{4,2}^{\mathsf{chal}}$. Then the protocol Π is $(\boldsymbol{K}, \boldsymbol{R}, \boldsymbol{\Phi})$-coordinate-wise predicate-special-sound for the primary LaBRADOR relation R_σ with $\sigma = \sqrt{\lambda/C_2}$.*

Proof. Given a $t \in \mathbb{T}_1$ for a statement $(\mathcal{F}, \mathcal{F}', \beta)$ such that $\boldsymbol{\Phi}(t) = 1$, we know from Φ_1^{prop}, Φ_2^{prop} and Φ_3^{prop} that a witness can be computed from t such that $((\mathcal{F}, \mathcal{F}', \beta), (\vec{\boldsymbol{w}}_1^*, \dots, \vec{\boldsymbol{w}}_r^*)) \in R_\sigma$. □

I.3 Binding Relation

In the first round LaBRADOR has both inner and outer commitments. Inner commitments are all produced with respect to $\boldsymbol{A} \in \mathcal{R}_{q'}^{\kappa\times n}$. The outer commitments are the sum of commitments using the matrices $\boldsymbol{B}_i \in \mathcal{R}_{q'}^{\kappa_1\times\kappa}$ for $i \in [1, r]$ and $\boldsymbol{C}_{ijk} \in \mathcal{R}_{q'}^{\kappa_2\times 1}$ for $i \in [1, r]$, $j \in [i, r]$, $k \in [0, t_2 - 1]$. The norms of openings to outer commitments are only ever checked together, therefore, when $\kappa_1 = \kappa_2$ we may therefore treat the commitment as one large Ajtai commitment for a matrix $\boldsymbol{B}$. Similarly, for the second outer commitment we may consider $\boldsymbol{D}$ rather than $\boldsymbol{D}_{i,j,k} \in \mathcal{R}_{q'}^{\kappa_2\times 1}$ for $i \in [1, r]$, $j \in [i, r]$, $k \in [0, t_1 - 1]$. For $n_1 = \left(r\kappa + \frac{r^2+r}{2}t_2\right)$ and $n_2 = \left(\frac{r^2+r}{2}t_1\right)$, $\boldsymbol{B} \in \mathcal{R}_q^{\kappa_1\times n_1}$, $\boldsymbol{D} \in \mathcal{R}_q^{\kappa_1\times n_2}$. In the case where binding is broken we wish to extract solutions to the Module-SIS problem, in particular we consider the relation

$$R_{\kappa,n,\boldsymbol{A},\beta^*}^{\mathsf{M\text{-}SIS}} = \{\; \vec{\boldsymbol{v}} \mid \vec{\boldsymbol{v}} \in \mathcal{R}_q^n,\; \boldsymbol{A} \cdot \vec{\boldsymbol{v}} = \vec{\mathbf{0}},\; 0 < \|\vec{\boldsymbol{v}}\| \le \beta^* \}.$$

Lemma I.6. *The predicate system $\boldsymbol{\Phi}$ admits*

$$R_{\mathsf{pp}}^{\mathsf{M\text{-}SIS}} = R_{\kappa,n,8T_{\mathrm{op}}(b+1)\beta',\mathsf{pp}.\boldsymbol{A}}^{\mathsf{M\text{-}SIS}} \cup R_{\kappa_1,n_1,2\beta',\mathsf{pp}.\boldsymbol{B}}^{\mathsf{M\text{-}SIS}} \cup R_{\kappa_1,n_2,2\beta',\mathsf{pp}.\boldsymbol{D}}^{\mathsf{M\text{-}SIS}}$$

as a binding relation.

Proof. First, we consider $\mathit{\Phi}_3^{\mathsf{bind}}(t_4^1, t_4^2)$ where $\boldsymbol{\Phi}(t_4^i) = 1$ for $i = 1, 2$. Recall for the predicate to be satisfied the following conditions must hold

$$(\vec{\boldsymbol{w}}_{i,1}^*, \bar{\boldsymbol{c}}_{i,1})_{i\in[r]} \leftarrow E_4(t_4^1),\ (\vec{\boldsymbol{w}}_{i,2}^*, \bar{\boldsymbol{c}}_{i,2})_{i\in[r]} \leftarrow E_4(t_4^2)$$
$$\forall i \in [r], \vec{\boldsymbol{w}}_{i,1}^* = \vec{\boldsymbol{w}}_{i,2}^*,\ \vec{\boldsymbol{v}}_{i,1} = \vec{\boldsymbol{v}}_{i,2}.$$

If the predicate is violated there must be some $i \in [r]$ such that $\vec{\boldsymbol{w}}_{i,1}^* \neq \vec{\boldsymbol{w}}_{i,2}^*$ or $\vec{\boldsymbol{v}}_{i,1} \neq \vec{\boldsymbol{v}}_{i,2}$.

Let us first address the case where $\vec{\boldsymbol{w}}_{i,1}^* \neq \vec{\boldsymbol{w}}_{i,2}^*$ but $\vec{\boldsymbol{v}}_{j,1} = \vec{\boldsymbol{v}}_{j,2}$ for all $j \in [r]$. In this case we have a two distinct weak openings for $\vec{\boldsymbol{v}}_{i,1} = \vec{\boldsymbol{v}}_{i,2}$, allowing us to find a short solution $\vec{\boldsymbol{v}} = \bar{\boldsymbol{c}}_{i,1}\bar{\boldsymbol{c}}_{i,2}(\vec{\boldsymbol{w}}_{i,1}^* - \vec{\boldsymbol{w}}_{i,2}^*)$. This may be seen as,

$$\boldsymbol{A}\vec{\boldsymbol{w}}_{i,1}^* = \vec{\boldsymbol{v}}_{i,1} = \boldsymbol{A}\vec{\boldsymbol{w}}_{i,2}^*,$$

giving $\boldsymbol{A}(\vec{\boldsymbol{w}}_{i,1}^* - \vec{\boldsymbol{w}}_{i,2}^*) = \vec{\boldsymbol{0}}$, which in turn implies $\boldsymbol{A}\bar{\boldsymbol{c}}_{i,1}\bar{\boldsymbol{c}}_{i,2}(\vec{\boldsymbol{w}}_{i,1}^* - \vec{\boldsymbol{w}}_{i,2}^*) = \vec{\boldsymbol{0}}$. This solution is indeed short as, when $\boldsymbol{\Phi}(t_4^1) = \boldsymbol{\Phi}(t_4^2) = 1$ then

$$\begin{aligned}\left\|\bar{\boldsymbol{c}}_{i,1}\bar{\boldsymbol{c}}_{i,2}(\vec{\boldsymbol{w}}_{i,1}^* - \vec{\boldsymbol{w}}_{i,2}^*)\right\|_2 &\leq \left\|\bar{\boldsymbol{c}}_{i,2}(\bar{\boldsymbol{c}}_{i,1}\vec{\boldsymbol{w}}_{i,1}^*)\right\|_2 + \left\|\bar{\boldsymbol{c}}_{i,1}(\bar{\boldsymbol{c}}_{i,2}\vec{\boldsymbol{w}}_{i,2}^*)\right\|_2 \\ &\leq 2T_{\mathrm{op}}\left\|\bar{\boldsymbol{c}}_{i,1}\vec{\boldsymbol{w}}_{i,1}^*\right\|_2 + 2T_{\mathrm{op}}\left\|\bar{\boldsymbol{c}}_{i,2}\vec{\boldsymbol{w}}_{i,2}^*\right\|_2 \leq 4T_{\mathrm{op}}(2(b+1)\beta').\end{aligned}$$

Now we must consider the case where $\vec{\boldsymbol{v}}_{i,1} \neq \vec{\boldsymbol{v}}_{i,2}$ for some $i \in [r]$. In this case we obtain two distinct openings with norm bound β' for a commitment using the matrix $\boldsymbol{B}$, from which we may compute a solution of norm $2\beta'$. The analysis for $\mathit{\Phi}_3^{\mathsf{bind}}$ may be applied to the predicates $\mathit{\Phi}_1^{\mathsf{bind}}, \mathit{\Phi}_2^{\mathsf{bind}}$.

If $\mathit{\Phi}_4^{\mathsf{bind}}$ is violated there must either be distinct openings to the outer commitments or distinct $\vec{\boldsymbol{y}}_{i,j} = \vec{\boldsymbol{z}} - \boldsymbol{c}_i\vec{\boldsymbol{w}}_i^* - \boldsymbol{c}_j\vec{\boldsymbol{w}}_j^*$ values. Note that for the transcripts collected for the i-th coordinate,

$$\vec{\boldsymbol{y}}_i(t) \neq \vec{\boldsymbol{y}}_i(t') \Rightarrow \vec{\boldsymbol{y}}_i(t) - \boldsymbol{c}_{1,j}\vec{\boldsymbol{w}}_j^* \neq \vec{\boldsymbol{y}}_i(t') - \boldsymbol{c}_{1,j}\vec{\boldsymbol{w}}_j^* \Rightarrow \vec{\boldsymbol{y}}_{i,j}(t) \neq \vec{\boldsymbol{y}}_{i,j}(t'),$$

so the $\vec{\boldsymbol{y}}_i$ are already covered by the $\vec{\boldsymbol{y}}_{i,j}$ case. Assume two subtrees t and t' extracted for the same coordinate i have $\vec{\boldsymbol{y}}_{i,j} \neq \vec{\boldsymbol{y}}'_{i,j}$ for some $j \in [r] \setminus \{i\}$. By the verification check, $\boldsymbol{A}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}') = (\boldsymbol{c}_i - \boldsymbol{c}'_i)\vec{\boldsymbol{v}}_i$, giving

$$\boldsymbol{A}(\vec{\boldsymbol{y}}_{i,j} + \boldsymbol{c}'_i\vec{\boldsymbol{w}}_i^* - \vec{\boldsymbol{y}}'_{i,j} - \boldsymbol{c}_i\vec{\boldsymbol{w}}_i^*) = (\boldsymbol{c}_i - \boldsymbol{c}'_i)\vec{\boldsymbol{v}}_i.$$

We know $\vec{\boldsymbol{w}}_i^*$ and $\bar{\boldsymbol{c}}_i$ make up a weak opening, so we have $\vec{\boldsymbol{v}} = \bar{\boldsymbol{c}}_i(\vec{\boldsymbol{y}}_{i,j} - \vec{\boldsymbol{y}}'_{i,j})$ where $\boldsymbol{A}\vec{\boldsymbol{v}} = \boldsymbol{0}$ and $\|\vec{\boldsymbol{v}}\|_2 \leq 8T_{\mathrm{op}}(b+1)\beta'$.

Consider an outer commitment with two distinct openings, i.e. we have trees t, t' where for some i, j one of the following hold

$$\vec{\boldsymbol{v}}_i(t) \neq \vec{\boldsymbol{v}}_i(t'), \quad \boldsymbol{g}_{i,j}(t) \neq \boldsymbol{g}_{i,j}(t'), \quad \boldsymbol{h}_{i,j}(t) \neq \boldsymbol{h}_{i,j}(t').$$

By verification, these are openings of norm less than β' for matrix $\boldsymbol{B}$ or $\boldsymbol{D}$, giving a solution of norm at most $2\beta'$. ◻

I.4 Knowledge Soundness

Combining our results we conclude that LaBRADOR is knowledge sound.

Theorem I.1. *Let Π be the base LaBRADOR protocol as described in Protocol 2 and 3. We consider the case with a ring $\mathcal{R}_q$ of degree d with splitting factor l and a B-well-spread challenge set, where each challenge has operator norm at most T_{op}. Restrict statements in the LaBRADOR relation to have ℓ_2-norm bound β at most q/C_1, where C_1, C_2 are the parameters of Lemma 2.2. The Fiat-Shamir transformation of LaBRADOR is knowledge sound for the relation $R_\sigma \cup R_{\mathsf{pp}}^{\mathsf{M\text{-}SIS}}$ for $\sigma = \sqrt{\lambda/C_2}$ with knowledge error* [12]

$$2(Q+1)(2^{-\lambda} + q^{-\lceil \lambda/\log q\rceil} + q^{-d/l} + (5+2l)rB),$$

where the extractor in expectation makes at most $K + Q(K-1)$ queries to the prover for $K = 2^3 \cdot (2r+1)$ and the prover makes at most Q oracle queries.

Proof. This follows by Theorem G.1 and Lemmata I.1 to I.6.

[12] For the sake of simplicity we assume the failure density terms dominate $(k_i - 1)/|\mathcal{C}_i|$ for each round i.

Remark I.1. To plug in Fiat-Shamir LaBRADOR into Lemma C.2 to realize an aggregate signature scheme, we must make sure that it also satisfies $\mathcal{Z}$-auxiliary input knowledge soundness (Definition B.12). It is easy to see that the existence of $\mathsf{aux\text{-}in}$ does not interfere with extraction at all. Observe that the distribution of $\mathsf{aux\text{-}in}$ output by $\mathcal{Z}(1^\lambda)$ is independent of any random oracle responses in the abstract sampling game, because it merely consists of a sequence of random preimage-image pairs and a public key of the underlying hash-then-sign signature scheme. Since $\mathsf{aux\text{-}in}$ is already fixed before the abstract sampling game starts, the above analysis goes through without modification even in the presence of $\mathsf{aux\text{-}in}$, by considering a prover with their input tape filled with $\mathsf{aux\text{-}in}$ as another fixed cheating prover with no auxiliary input. Note also that $\mathsf{aux\text{-}in}$ is independent of the LaBRADOR public parameters pp_Π, which allows one to construct a reduction solving $\mathsf{M\text{-}SIS}$ without any problem whenever a $\mathsf{M\text{-}SIS}$ instance has to be embedded in pp_Π.

I.5 Recursive Composition

Let us now consider recursively composing the LaBRADOR protocol t times. This means that the prover no longer responds with $\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \vec{\boldsymbol{g}}, \vec{\boldsymbol{h}}$ after the first iteration, but instead commits to it as the first message of the next iteration, only leaving the opening of the last iteration.

For a single LaBRADOR iteration an accepting transcript directly guarantees that the checks performed by an honest verifier hold, thus the guarantees for $\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \vec{\boldsymbol{g}}, \vec{\boldsymbol{h}}$ must now instead be enforced by the predicates on the subsequent iteration. We may see a sequence of t compositions as $(\boldsymbol{K}, \boldsymbol{R}, \boldsymbol{\Phi})$-coordinate-wise predicate-special-sound, for

$$\boldsymbol{K} = (2, 2, 2, 3, \ldots, 2, 2, 2, 3), \;\; \boldsymbol{R} = (1, 1, 1, r_1, \ldots, 1, 1, 1, r_t).$$

The predicate system $\boldsymbol{\Phi}$ must now be obtained by applying the same predicates as for a single iteration repeatedly, going left to right starting from the first level:

$$\Phi_1^{\mathsf{com}}, \Phi_2^{\mathsf{com}}, \Phi_3^{\mathsf{com}}, (\Phi_{4,1}^{\mathsf{chal}}, \Phi_{4,2}^{\mathsf{chal}}, \Phi_4^{\mathsf{com}}), \ldots, \Phi_1^{\mathsf{com}}, \Phi_2^{\mathsf{com}}, \Phi_3^{\mathsf{com}}, (\Phi_{4,1}^{\mathsf{chal}}, \Phi_{4,2}^{\mathsf{chal}}, \Phi_4^{\mathsf{com}}).$$

Note the norm checks enforced by Φ_1^{com} all have slack $\sqrt{\lambda / C_2}$. Thus, the norms of the $\mathsf{M\text{-}SIS}$ must tolerate this additional slack, too. The soundness error now accumulates additively for each iteration,

$$2(Q+1) \sum_{i=1}^{t} \left(2^{-\lambda} + q^{-\lceil \lambda / \log q \rceil} + q^{-d/l} + (5 + 2l) r_i B \right), \tag{18}$$

while the runtime composes multiplicatively, giving $K + Q(K-1)$ queries to the prover for $K = \prod_{i=1}^{t} (2^3 \cdot (2r_i + 1))$. Recall, the sequence $r_1, \ldots, r_t$ is exponentially decreasing.

I.6 Last Iteration Optimizations

In its original presentation, the final iteration of LaBRADOR was optimized to reduce the proof size; as the final message is not given as input to another iteration the garbage polynomials may instead be distributed across r prover messages, where r is the number of witness vectors. Using techniques from [NS22], this allows garbage polynomials to depend on previous challenges, giving a smaller number of polynomials in total. Previously we have used 3-special-sound trees to extract witnesses and perform Schwartz-Zippel checks. Extending this approach naïvely, extracting one witness for each level in the tree quickly proves problematic, when considering the norms of the extracted openings. For each new witness the norm of its weak opening will depend on the norms of all previous openings with an extra factor depending on the operator norm. This causes the norms of the weak openings to have slack proportional to $(T_{\mathrm{op}})^r$. To ensure binding still holds for weak openings with this added slack, the rank of the commitments must grow significantly. The size impact of larger commitments quickly outweighs the benefits of the optimization as r grows.

In the interactive setting it was possible to extract each witness $\vec{\boldsymbol{w}}_i^*$ independently, by finding transcripts where the challenge only differs in the ith round. However, it seems unclear how such an extraction strategy would work in the non-interactive setting. One alternative solution could be adding a final round with an extra amortized opening for new challenges. Witnesses could then be extracted from this new opening as in a normal LaBRADOR iteration, and used to enforce binding with respect to the other opening, avoiding the exponential slack in r. The second opening may outweigh the concrete gains made by applying the optimization.